METODI E SOFTWARE PER LA CODIFICA AUTOMATICA E ASSISTITA DEI DATI

COLLANA: TECNICHE E STRUMENTI**, N. 4 - 2007**

***Capitoli redatti da**:*

Introduzione	Stefania Macchia
Cap. 1	
Par. 1.1, 1.2, 1.3 , 1.4.1	Stefania Macchia
Par. 1.4.2	Paola Giovani
Cap. 2	
Par. 2.1, 2.2, 2.3	Stefania Macchia
Par. 2.4, 2.5	Domenico Perrone
Cap. 3	Massimiliano Degortes
Cap. 4	Paola Giovani
Cap. 5	
Par. 5.1, 5.2	Stefania Macchia
Par. 5.3, 5.4	Domenico Perrone
Cap. 6	Loredana Mazza

Per informazioni rivolgersi a: Segreteria MTS - Istat 06/46732320
Giulio Barcaroli - responsabile Servizio MTS
Stefania Macchia - responsabile U.O. MTS/H

Indice

Introduzione: cosa contiene questo volume e come utilizzarlo

Questo volume è articolato in sei capitoli, che introducono progressivamente alla problematica della codifica con l'ausilio del computer, fino a trarre spunto da applicazioni già sviluppate per fornire indicazioni su un utilizzo ottimale dello specifico sistema di codifica automatica o assistita.

In particolare, nel primo capitolo si forniscono cenni metodologici sulla tematica della codifica di risposte a testo libero fornite nel corso di indagini statistiche, evidenziando le motivazioni che inducono a preferire un approccio automatizzato, citando le esperienze effettuate da altri istituti nazionali di statistica, nonché delineando le possibili strategie da adottare nel corso della gestione di un'indagine.

I capitoli che seguono rappresentano invece i manuali veri e propri dei sistemi di codifica che costituiscono ormai due standard in Istat, rispettivamente per la codifica automatica ed assistita: Actr e Blaise.

Il secondo capitolo, in particolare, è il manuale del sistema di codifica Actr; questo manuale, rispetto a quello prodotto da Statistics Canada, è stato riorganizzato in modo da evidenziare ed anteporre la parte più delicata inerente la standardizzazione dei testi ed in modo da far riferimento ad esempi tipici di variabili rilevate nel corso di indagini condotte presso l'Istat.

Nel terzo capitolo viene descritto un software di corredo ad Actr sviluppato in Istat finalizzato a facilitare l'aggiornamento delle applicazioni di codifica automatica, tramite l'individuazione di eventuali incoerenze che possono venirsi a creare, soprattutto in caso di classificazioni molto complesse.

Nel quarto capitolo, che costituisce la guida all'utilizzo del sistema di codifica Actr, si fa riferimento ad una applicazione particolarmente complessa già sviluppata, quella relativa alla Classificazione delle attività economiche, per dare indicazioni comportamentali per chi si trovi a dover sviluppare ex novo una procedura di codifica e per mettere in luce gli aspetti più delicati che devono essere costantemente tenuti sotto controllo.

I capitoli cinque e sei sono infine dedicati al sistema Blaise, limitatamente al modulo di codifica assistita ed, in particolare, come per Actr, il quinto capitolo costituisce il manuale utente, mentre il sesto, la guida all'utilizzo del sistema.

1 Cenni metodologici: l'automazione del processo di codifica

1.1 Perché automatizzare il processo di codifica

L'attività di codifica delle risposte ai quesiti a testo libero, nell'ambito del processo di gestione di indagini statistiche, costituisce un momento estremamente delicato e che ha un forte impatto sulla qualità dei dati. Volendo dare una definizione, possiamo dire che per *codifica* si intende *ricondurre risposte testuali spontanee a classificazioni predefinite*.
Per esempio:

```
Qual è il suo comune di nascita?
   Castiglion del Lago    (classificazione dei comuni Italiani
2001)

Descriva la sua professione
   Impiegato    amministrativo    addetto    alle    fotocopie
(classificazione Istat delle Professioni 2001).
```

Se realizzata manualmente, tale attività, apparentemente banale, comporta una serie di problematiche riconducibili a:

- onerosità in termini di tempo;
- onerosità in termini di risorse;
- standard qualitativi non sempre elevati.

I fattori che maggiormente incidono su queste problematiche sono la modalità di rilevazione (tradizionale, tramite questionario cartaceo, oppure con il supporto del computer), la complessità della classificazione (numero di voci, sistematicità dei criteri classificatori), nonché l'esperienza e la competenza specifica dei codificatori.
Le rilevazioni censuarie costituiscono l'esempio più emblematico di difficoltà inerenti l'attività di codifica, in quanto:

- la mole di dati da codificare è estremamente ampia;
- il numero di addetti da dedicare a questa attività è molto elevato;
- ne consegue che i costi ed i tempi di addestramento sulle diverse classificazioni siano notevoli e non sia sempre possibile garantire livelli di competenza omogenei.

Relativamente agli standard qualitativi non elevati della codifica manuale, è stato dimostrato che, in caso di classificazioni complesse, la possibilità che due codificatori diversi attribuiscano codici diversi a risposte testuali identiche si verifica più spesso di quanto si possa pensare. Infatti, da uno studio effettuato su un campione di dati del Censimento della popolazione del 1991, reintervistato a distanza di tre mesi in occasione dell'Indagine di qualità del censimento stesso, è emerso che, relativamente alle Classificazioni delle professioni e dei titoli di studio, la divergenza tra codici assegnati da persone diverse (addetti al Censimento prima ed all'Indagine di qualità poi) fosse

rispettivamente pari al 41 per cento nel caso della professione ed al 30 per cento per il titolo di studio.[1]
Questi risultati non stupiscono se si pensa che, nella gran parte dei casi, le risposte fornite dagli intervistati non sono immediatamente riconducibili ai codici delle classificazioni di riferimento, ma richiedono sforzi interpretativi da parte di ciascun codificatore, dando quindi adito a risultati diversi.
L'adozione di strumenti automatici consente di definire una volta per tutte:

- i criteri di prevalenza da adottarsi nel caso di risposte con contenuti multipli afferenti ciascuno a codici diversi;
- le sfumature interpretative adottate nel linguaggio parlato, che possono sottintendere significati diversi;
- le modalità di trattamento di risposte non sufficientemente informative per l'attribuzione di codici al massimo livello di dettaglio.

Per tutti questi motivi, molti Istituti nazionali di statistica hanno investito nella ricerca di sistemi *software* finalizzati all'automazione del processo di codifica delle risposte a testo libero.
La *performance* dei diversi sistemi adottati varia molto al variare del livello di complessità delle classificazioni considerate e delle modalità di rilevazione. Dalle esperienze effettuate negli altri paesi europei ed extraeuropei risulta che in generale i tassi d'accoppiamento (risposta empirica – assegnazione di un singolo codice) ottenuti per classificazioni semplici vanno dal 70 per cento al 90 per cento, mentre, per classificazioni più complesse (tipicamente Attività economica e Professione), si rilevano valori che variano dal 60 per cento all'80 per cento, con percentuali più basse quando i dati codificati sono censuari, come riportato nella seguente tavola.[2]

Tavola 1 - Esperienze di codifica automatica

Paese	***Ateco %***	***Professione %***
Usa (1984 Indagine sui consumi delle famiglie)	78	74
Usa (1990 Benchmark file del Censimento)	74	61
Danimarca (Modulo per la compilazione delle tasse e Indagine sulle forze di lavoro)		75
Australia (1989 Test di Codifica automatica)		70
Svezia (1980 1985 Censimenti)	61	71
Francia (1990 Censimento)		66
Croazia (1991 Censimento)	64	66
Francia (1995 Indagine sulle forze di lavoro)		80

Al di là dei singoli risultati, è significativo osservare che nessuno dei paesi che abbiano utilizzato sistemi di codifica automatica è mai tornato indietro alla codifica manuale; al contrario, anche in caso di risultati non soddisfacenti, si è preferito investire ulteriormente sulla codifica automatica (ad

[1] De Angelis Roberta, e Stefania Macchia, e Loredana Mazza. A*pplicazioni sperimentali della codifica automatica: analisi di qualità e confronto con la codifica manuale.* Roma: Istat, 2000. (Quaderni di ricerca – Rivista di statistica ufficiale, n.1, 29-54).
[2] Lyberg and Dean. *Automated Coding of Survey Responses: an international review*. Washington: DC 1992. (Conference of european statisticians, work session on Statistical data editing).
Kalpic D. *Automated coding of census data*. 1994. (Journal of Official Statistics, vol. 10, 449-463).
Touringny J.Y. and Moloney. *The 1991 Canadian Census of Population experience with automated coding, statistical data editing, 2* New York: 1995. (United nations statistical commission).

esempio gli Usa, dopo i risultati non ottimali in termini d'accuratezza ottenuti sul *benchmark file* del 1990, hanno ulteriormente sviluppato i loro sistemi di codifica e li hanno applicati successivamente al Censimento dell'Industria).

1.2 La definizione della strategia di codifica

Si premette che il supporto di *software* specifici nella fase di codifica di quesiti rilevati a testo libero può avvenire secondo due modalità:

- **codifica automatica** (**Auc**– **Au**tomatic **C**oding) → il software analizza in batch un file contenente l'insieme di risposte testuali;
- **codifica assistita** (**Cac** – **C**omputer **A**ssisted **C**oding)→ il software costituisce un supporto al codificatore, facilitandone la navigazione nel dizionario della classificazione di riferimento.

È evidente quindi che, mentre secondo la prima modalità si delega al software il compito di *prendere autonomamente una decisione* sul codice da attribuire ad ogni singolo testo, nel caso della codifica assistita ci si avvale del codificatore per la conferma del codice individuato dal software o per la selezione di un codice tra quelli proposti da questo stesso.

Ne consegue che gli obiettivi propri di ciascuna di queste due modalità operative siano diversi:

- nel caso della **codifica automatica** la finalità è di individuare ed estrarre dal dizionario **una singola** descrizione che realizzi il match con quella da codificare;
- nel caso della **codifica assistita**, il software costituisce un supporto interattivo al codificatore, facilitandone **la navigazione nel dizionario** della classificazione di riferimento. Quindi può essere opportuno estrarre dal dizionario un set di descrizioni, anche molto simili tra loro, lasciando al codificatore la selezione di quella corretta.

Ne consegue che, mentre ci si può attendere che un sistema di codifica assistita sia in grado di attribuire un codice alla totalità delle risposte aventi un contenuto informativo sufficiente per essere codificate, un sistema di codifica automatica molto difficilmente codificherà la totalità di risposte codificabili, in quanto ne lascerà necessariamente senza codice una percentuale per la quale la *decisione* sul codice da attribuire non è così scontata da soddisfare i criteri logici in base ai quali è stato sviluppato.

Nel processo di gestione di un'indagine, la definizione della **strategia di codifica**, consistente nell'adottare l'una modalità, l'altra, oppure un mix tra le due, può dipendere da una serie di fattori, quali:

- la tecnica di rilevazione:
  - assistita da computer con rilevatore (Cati - *Computer Assisted Telephone Interviews* -, Capi - *Computer Assisted Personal Interviews*);
  - assistita da computer senza rilevatore (Casi – *Computer Assisted Self Interviews*);
  - questionario cartaceo (Papi - *Paper and Pencil Interviews*)
- la mole di dati da codificare;
- la lunghezza dell'intervista;
- la struttura della classificazione.

In linea generale, è evidente che, mentre la codifica automatica può essere realizzata soltanto dopo la cattura dei dati, la codifica assistita può essere adottata sia in fase di rilevazione che successivamente.

È chiaro, però, che la codifica assistita in fase di rilevazione è fattibile soltanto nel caso di tecnica di acquisizione assistita da computer; d'altro canto, soprattutto nel caso di tecnica Casi, la codifica in fase di intervista è pensabile soltanto nel caso che la classificazione di riferimento sia estremamente semplice.
La lunghezza dell'intervista, d'altro canto, può essere un deterrente della codifica in fase di rilevazione, in quanto, soprattutto in caso di classificazioni complesse, questa potrebbe contribuire ad appesantire l'intervista ed a comprometterne la conclusione con esito positivo.
La mole di dati da trattare, invece, potrebbe costituire un incentivo all'adozione della codifica automatica, in funzione del risparmio ottenibile in termini di tempo e di risorse da dedicare; in tal caso, soprattutto nell'eventualità di classificazioni complesse, si può pensare ad una prima fase di codifica automatica ed a una seconda di codifica assistita, per risolvere i casi lasciati aperti dalla prima.
I tre schemi che seguono riassumono le strategie di codifica ottimali in funzione dei fattori precedentemente citati.

Schema 1 - Codifica in fase di rilevazione con operatore

	Lunghezza dell'intervista	
Struttura della classificazione	**Breve**	**Lunga**
• **Semplice**	CAC	CAC
• **Complessa & grande variabilità delle risposte**	CAC	- (codifica successiva alla rilevazione)

Schema 2 - Codifica in fase di rilevazione senza operatore

Struttura della classificazione	
Semplice	CAC
Complessa & grande variabilità delle risposte	- (codifica successiva alla rilevazione)

Schema 3 - Codifica in fase successiva alla rilevazione

	Mole di dati	
Struttura della classificazione	**Large number**	**Small number**
Semplice	AUC + CAC	AUC
Complessa & grande variabilità delle risposte	AUC + CAC	CAC

1.3 I sistemi di codifica automatica/assistita

Le fasi di lavorazione principali previste da qualunque sistema di codifica automatica/assistita includono:

- la costruzione di un dizionario informatizzato;
- la registrazione su supporto informatico delle risposte testuali da codificare,
- la standardizzazione delle descrizioni testuali,
- la codifica vera e propria attraverso il matching tra descrizioni-risposta e descrizioni del dizionario,
- il controllo di qualità della codifica automatica/assistita.

1.3.1 Costruzione del dizionario

La base per la costruzione di un dizionario informatizzato è il manuale ufficiale della classificazione di riferimento; questo ultimo, però, per poter essere trattato da un software, deve essere sottoposto ad una serie di operazioni dovute al fatto che le descrizioni testuali delle classificazioni ufficiali sono state pensate per essere utilizzate dai codificatori, che possono fare deduzioni o riferirsi alle loro conoscenze personali. Dovendo invece affidarsi ad un software, è necessario fornire a quest'ultimo tutti gli elementi ai quali farebbe automaticamente riferimento una mente umana. Bisogna quindi effettuare una serie di operazioni finalizzate ad includere nei dizionari processabili esclusivamente descrizioni sintetiche, analitiche e non ambigue.
È necessario in pratica: semplificare le descrizioni, definire i sinonimi, rielaborare le classi aperte, eliminare le clausole di escluso e integrare i dizionari con materiale di riferimento.
Questi passaggi sono assolutamente imprescindibili nel caso di applicazioni di codifica automatica, nell'ambito delle quali l'individuazione del codice da assegnare è completamente delegata al software; nel caso della codifica assistita, il dettaglio con cui realizzare le diverse fasi di seguito descritte è soprattutto funzionale al fatto di limitare al massimo la discrezionalità del processo di codifica, in quanto, invece, l'eventuale assenza di una molteplicità di sinonimi potrebbe essere superata dalla prontezza del codificatore di effettuare la ricerca testuale tramite vocaboli alternativi.

- **Semplificare le descrizioni** → spesso descrizioni che riassumono più di un concetto sono associate a singoli codici, mentre tipicamente il rispondente si riferisce a concetti singoli. Per esempio: la nostra Classificazione delle professioni assegna un unico codice a "`matematici e statistici`", mentre il rispondente potrebbe rispondere soltanto "`matematico`", oppure soltanto "`statistico`", a seconda della sua specializzazione. In casi come questo, è necessario

suddividere la frase in due o più descrizioni ed associarle allo stesso codice.

- **Definire i sinonimi** → in questo contesto, per sinonimi si intende sia l'equivalenza di parole che esprimono lo stesso concetto (ad es.: le parole "*scarpe"* e "*calzature"*), sia termini specifici che la classificazione in oggetto riconduce a un termine generico. Infatti, le classificazioni contengono spesso parole generiche che si riferiscono a categorie, mentre i rispondenti possono utilizzare parole specifiche. Per esempio: al posto di "agricoltore di cereali", il rispondente potrebbe rispondere "agricoltore di grano". È quindi necessario esplicitare la lista dei termini puntuali riconducibili al concetto generico.
- **Rielaborare le classi aperte** → abitualmente le classificazioni prevedono descrizioni del tipo "Altri...", intendendo "*qualcos'altro rispetto ai concetti già specificati*". Per esempio, "Altri operai specializzati", laddove, differenti tipi di operai specializzati sono già stati elencati in precedenza. Si deve quindi, per quanto possibile, elencare tutti i concetti ai quali le classi aperte si riferiscono.
- **Eliminare le clausole di escluso** → chiaramente un sistema di codifica automatica non può ragionare in termini di esclusione, quindi non può capire il significato di "escluso…", "a parte…*"* ed altre clausole affini che escludono determinate categorie da alcune classi. Bisogna quindi eliminare queste clausole dai dizionari e verificare che i concetti "esclusi" da una classe siano inclusi in un'altra.
- **Integrare i dizionari con materiale di riferimento** → i dizionari processabili possono essere ampliati con descrizioni provenienti da note esplicative degli stessi manuali delle classificazioni oppure da altre classificazioni correlate. Per esempio, ogni volta che la classificazione delle Attività economiche cita un elemento riguardante la produzione di una *"certa categoria di prodotti"* (che riassume una lista implicita) può essere utilizzata la classificazione dei prodotti per elencare esplicitamente la lista di prodotti.

1.3.2 Registrazione su supporto informatico delle risposte testuali da codificare

È banale specificare che per utilizzare un sistema di codifica automatica/assistita è necessaria la registrazione delle risposte testuali da codificare su supporto informatico.
Tralasciando il caso delle tecniche di rilevazione assistite da computer (Capi – Computer Assisted Personal Interviewing e Cati - Computer Assisted Telephone Interviewing), è invece opportuno fare qualche cenno sulla tecnica d'acquisizione utilizzata. Mentre, infatti, la registrazione tradizionale garantisce livelli di qualità ormai consolidati anche per le variabili alfabetiche, altrettanto non può essere detto per la lettura ottica. È dimostrato, infatti, che con questa tecnologia le percentuali di riconoscimento dei caratteri alfabetici difficilmente superano valori del 80-90 per cento. È consigliabile quindi demandare l'acquisizione delle variabili alfabetiche ad un processo di *"keying from images"*, riportandosi così ad una situazione assimilabile alla registrazione tradizionale.
Per entrambe le tecniche, comunque, è preferibile dotarsi di un correttore ortografico, in modo da minimizzare gli errori ortografici che possono avere effetti negativi sul processo di codifica automatica.

1.3.3 Standardizzazione delle descrizioni testuali

I sistemi di codifica automatica usualmente fanno precedere alla fase di codifica vera e propria una elaborazione attraverso la quale le descrizioni testuali vengono ridotte ad una forma *"canonica"* dipendente dalla lingua, dallo specifico dominio dell'applicazione e dal particolare sistema di codifica utilizzato. Scopo di questa elaborazione, chiamata "*parsing"*, è quello di eliminare dal testo la variabilità grammaticale o sintattica che non incide sull'aspetto semantico, ma soltanto sulla forma, e che pertanto è irrilevante ai fini dell'abbinamento con le voci del dizionario. Processi di standardizzazione tipici sono: la mappatura dei caratteri, l'eliminazione delle parole ininfluenti, l'eliminazione dei prefissi e dei suffissi, il trattamento dei sinonimi.

1.3.4 Matching tra descrizioni-risposta e descrizioni del dizionario

I sistemi di codifica assistita da computer sono generalmente in grado di gestire due tipi di matching*:* uno definito *"esatto"*, l'altro definito *"parziale"*.

- Nel tipo di matching **esatto** → la descrizione da codificare, una volta standardizzata, è identica ad una di quelle contenute nel dizionario. Questo tipo di regola di ricerca, per quanto grezzo, può dare percentuali di codifica molto soddisfacenti nel caso di classificazioni non troppo articolate o di descrizioni costituite prevalentemente da parole singole.
- Nel tipo di matching **parziale** → se la descrizione da codificare non si accoppia perfettamente con una descrizione del dizionario, si ricorre a tecniche di matching parziale, ovvero si ricerca la descrizione *"più simile"* secondo opportune misure di distanza tra i testi (di tipo probabilistico, empirico o basate su informazioni ausiliarie).

I sistemi di codifica automatica utilizzano algoritmi di ricerca di tipo differente. Tra i principali algoritmi di ricerca utilizzati, occorre ricordare il *"dictionary"*, lo *"weighting"*, il *"matching di sottostringhe"* ed infine i *"sistemi esperti"*.

- **Dictionary algorithms** → algoritmi di costruzione del dizionario per individuare accoppiamenti esatti (metodo grezzo) o accoppiamenti parziali sulla base di parole (o gruppi di parole chiave), ovvero parole particolarmente informative che determinano univocamente l'assegnazione del codice.
- **Weighting algorithms** → ricerca di match esatti o parziali sulla base di funzioni di similarità tra i testi, dove alle parole è attribuito un peso, empirico o probabilistico, proporzionale al loro grado di informatività.
- **Matching di sottostringhe** (bigrammi o trigrammi) → ricerca di match parziali basati sull'accoppiamento di sottostringhe di testo. In particolare, nei sistemi sviluppati dall'Insee (Institut National de la Statistique et des Etudes Economiques) il numero di bigrammi da confrontare è minimizzato perché il dizionario è organizzato, in fase di apprendimento, in alberi di ricerca ottimali, costruiti calcolando la posizione dei bigrammi più informativi
- **Sistemi esperti** → metodi mediati dalla ricerca sull'intelligenza artificiale, ad esempio *Memory Based Reasoning* (U.S. Bureau of Census) che utilizza un particolare processore per il calcolo parallelo (*Connection Machine System*).

1.3.5 Controllo di qualità della codifica

La qualità di un sistema di codifica può essere misurata attraverso i seguenti parametri:

- **efficacia** → percentuale di codici assegnati automaticamente,
- **accuratezza** → percentuale di codici corretti assegnati automaticamente,
- **efficienza** → tempo unitario di assegnazione del codice.

È evidente che l'obiettivo da porsi in fase d'implementazione di un sistema di codifica è quello di un corretto bilanciamento tra efficacia ed accuratezza, in quanto la massimizzazione di uno solo dei due parametri non può che provocare effetti negativi sull'altro.

1.4 I sistemi di codifica attualmente in uso in Istat

Sono di seguito descritti i due sistemi di codifica attualmente in uso presso l'Istat, Actr e Blaise, utilizzati rispettivamente per la codifica automatica e per quella assistita.

1.4.1 Actr

Il sistema Actr v3 (Automated Coding by Text Recognition), sviluppato e commercializzato da Statistics Canada, rientra tra i sistemi basati sui cosiddetti "*weighting algorithms"*. È un software per la codifica automatica di variabili testuali di tipo **generalizzato**, in altre parole **indipendente dalla classificazione considerata e dalla lingua** in cui sono espressi i testi. Il sistema gestisce applicazioni di codifica completamente automatiche (modalità batch) e, con l'ausilio di un'interfaccia grafica, permette di analizzare interattivamente la codifica di casi singoli. Actr è stato utilizzato in molteplici applicazioni nell'ambito della statistica ufficiale inclusi i censimenti.[3]
La logica di base del sistema è ispirata alla metodologia sviluppata originariamente presso US Census Bureau [4] ed utilizza degli algoritmi d'abbinamento di dati testuali messi a punto successivamente da ricercatori di Statistics Canada.[5]
Anche in Actr, come negli altri sistemi di codifica automatica, il confronto tra la risposta da codificare e le voci contenute nel dizionario della classificazione è preceduto dalla fase definita "*parsing"*; tale fase è completamente controllata dall'utente che ha il compito di adattarla al particolare contesto applicativo (lingua, classificazione, tipologia di rispondente). La peculiarità di Actr rispetto ad altri sistemi è che, mettendo a disposizione fino a 14 diverse funzioni di *parsing*, consente una notevole flessibilità e possibilità di personalizzazione del processo di standardizzazione.
Il dizionario della classificazione, è sottoposto al processo di *parsing* e caricato in un database di sistema, allo scopo di ottimizzare i tempi d'accesso e di ricerca. Gli eventuali duplicati ed i casi d'incongruenza (codici differenti associati alla stessa descrizione testuale) della classificazione sono segnalati ed esclusi dal database. In questa fase vengono anche calcolati i pesi (P) associati alle

[3] Touringny J.Y. and Moloney. *The 1991 Canadain Census of Population experience with automated coding, statistical data editing, 2.* New York: 1995 (United nations statistical commission).
[4] Appel M. e E. Hellerman. *Census Bureau Experience with Automated Industry and Occupation Coding.* In American Statistical Association, 1983. (Proceedings of Section on Survey Research Methods, 32-40)
[5] Wenzowski M.J. *A Generalised Automated Coding System.* ACTR (Survey Methodology, vol. 14: 299-308).

singole parole contenute nel dizionario, che sono proporzionali all'informatività della parola stessa, ovvero alla sua capacità di discriminare univocamente un codice:

$$P(W_i) = 1 - \log(Nw_i)/\log(N) \quad (1)$$

dove Nw_i è il numero di codici contenenti la parola i-esima (W_i) mentre N è il numero totale di codici contenuti nel database. Nella fase di codifica vera e propria la risposta testuale è confrontata con il database alla ricerca di un abbinamento esatto (*direct match*), ovvero del testo che abbia tutte le parole in comune con la risposta, che dà luogo inequivocabilmente all'assegnazione di un codice unico. Se il tentativo fallisce viene ricercato un abbinamento parziale (*indirect match*), in questo caso il *software* individua, tramite una misura della similarità tra testi (S) di tipo empirico, *"il codice"* o *"i codici"* del dizionario con descrizione più simile alla risposta fornita dal rispondente. Tale misura è funzione del numero di parole in comune e del loro grado d'informatività all'interno del dizionario di riferimento:

$$S= 10\,(a + 2b)/3 \qquad (2)$$

$$a = 2\,N_C/(N_R + N_D)$$

$$b = 2\Sigma_i\, P(W_i^C)/\,\{\Sigma j\, P(W_J^R) + \Sigma_i\, P(W_i^D)\}$$

dove N_C è il numero di parole in comune tra risposta e testo del dizionario, N_R e N_D rappresentano il numero totale di parole contenute rispettivamente nella risposta e nel testo del dizionario, mentre $P(W_i^C)$, $P(W_J^R)$ e $P(W_i^D)$ rappresentano i pesi definiti nella (1) rispettivamente della i-esima parola in comune (W_i^C), della j-esima parola contenuta nella risposta testuale (W_J^R) e della i-esima parola del dizionario. La misura di similarità espressa nella (2) assume valori compresi nell'intervallo [0,10] i cui estremi corrispondono ad un abbinamento testuale nullo (S=0) o ad un abbinamento esatto (S=10). Il sottoinsieme dei testi del database con almeno una parola in comune con la risposta vengono ordinati per misura S decrescente ($S_1 > S_2 > \ldots > S_n$). La regione di accettazione per la misura di similarità è data dalle relazioni (3) ed è costruita utilizzando tre parametri soglia: Smin, Smax e ΔS, che rappresentano rispettivamente le soglie minima e massima di accettazione, e la minima distanza richiesta tra testo a punteggio massimo (S_1) e successivo (S_2). I possibili risultati del sistema Actr si suddividono quindi in:

$S_1 > Smax$ e $(S_1 - S_2) > \Delta S$ (codice *unico*) (3a)

$S_1 > Smax$ e $(S_1 - S_2) \leq \Delta S$ (codici *multipli*) (3b)

$Smin < S_1 \leq Smax$ (codici *possibili*) (3c)

$S_1 \leq Smin$ (casi *falliti*) (3d)

Se è soddisfatta la condizione (3a) la voce del dizionario a punteggio massimo ($S_{1)}$ è dichiarata *"vincente"*, il codice che le è associato è unico e viene assegnato in modo completamente

automatico. I rimanenti casi necessitano, invece, della valutazione da parte di codificatori (o di programmi ausiliari) che selezionino il codice corretto tra quelli proposti dal sistema. I valori dei parametri soglia sono fissati dall'utente in funzione dei suoi obiettivi di qualità: valori alti elevano l'accuratezza (percentuale di codici unici corretti) dei risultati a scapito dell'efficacia (percentuale di codici unici assegnati); quindi la scelta dei valori ottimali si gioca sul bilanciamento tra questi due aspetti della qualità dei risultati.
Tra le funzionalità più avanzate del sistema Actr citiamo l'uso dei **campi filtro** (confrontare paragrafo 2.4.4.7) e dei **contesti multipli di codifica** (confrontare paragrafo 2.4.4.8). I campi filtro possono essere utilizzati per restringere la ricerca del codice all'interno di sottoinsiemi di classi specificate dall'utente. Il sistema Actr consente inoltre di gestire applicazioni in cui la ricerca del codice è effettuata su più ambienti (costituiti ciascuno da un dizionario e dai relativi *file di parsing*) analizzati in cascata (se la ricerca fallisce nel primo contesto si passa al secondo e così via). L'impiego di contesti multipli di codifica è particolarmente utile nella codifica di testi bilingui, ma si estende a tutte le situazioni in cui sia possibile e vantaggioso trattare con più metodologie alternative i casi non risolti dal sistema (ad esempio con dizionari ausiliari).
Ci sembra importante ribadire che Actr è un software generalizzato, in quanto si basa su un algoritmo d'abbinamento di stringhe testo che è indipendente dalla lingua in cui sono scritte e dalla variabile cui si riferiscono. L'applicazione di Actr richiede però all'utente un'attività di adattamento alla lingua e alla variabile da codificare. Infatti, le regole di *parsing* e il dizionario della classificazione, che sono parte integrante dell'applicazione di codifica, dipendono dalla grammatica e sintassi specifiche della lingua e dal particolare tipo di risposta da codificare. Questa attività di adattamento, denominata *"addestramento"* del sistema, consiste nell'individuare gli opportuni correttivi al dizionario o alla *strategia di parsing* per ridurre al minimo i casi di fallimento o di codice multiplo. La *performance* del sistema Actr non può quindi essere valutata a priori, essa è necessariamente vincolata al particolare contesto applicativo e alla qualità della fase di addestramento.

1.4.2 Blaise

Come approfondito nel capitolo 5, Blaise, sviluppato e commercializzato da Statistics Netherlands, è un sistema per l'acquisizione dati assistita da computer (Cai – Computer Assisted Interviewing), che prevede una apposita funzione di codifica assistita per le risposte testuali.
L'utente (il codificatore) può utilizzare questa funzione, pur non essendo a conoscenza delle regole di trattamento dei testi implementate nel software, tuttavia, si ritiene opportuno fornire uno schema generale del funzionamento dell'algoritmo di ricerca dei dati.

1.4.2.1 La parola in Blaise: il Trigram

In Blaise le parole sono scomposte in stringhe di tre caratteri definite Trigram. Sono considerati caratteri, oltre alle lettere dell'alfabeto, anche lo spazio che le precede e le segue. Tutti i Trigram sono ottenuti in base ad uno *shift* di 1 orientato da sinistra verso destra che avanza dal primo carattere-spazio che precede la parola fino all'ultimo carattere-spazio che la segue.
Quindi il primo Trigram, composto con le due lettere con cui la parola comincia, sarà costituito dalla sequenza "spazio-lettera-lettera", mentre l'ultimo, composto con le due lettere con cui la parola termina, sarà costituito dalla sequenza "lettera-lettera-spazio".

Sulla base di quanto detto, si può quindi quantificare che il numero dei Trigram (NrOfTrigram) è uguale al numero di lettere della stringa di ricerca (Search String).
Invece, la lunghezza (numero di posizioni) della Working String risulta da un numero variabile di caratteri-lettera + 2 numero costante di caratteri-spazio.
Ad esempio, la parola di 5 lettere "segno" avrà Ncaratteri = 5 e NrOfTrigram = 5 e corrisponderà a una Working String di lunghezza L = 5+2 = 7.
La scansione della parola in triplette crea un array di Trigram.
La parola "segno" dell'esempio precedente corrisponderà ad un array T(1,5) dei W(7) caratteri della Working String:

$T_1 = W_1 + W_2 + W_3$
$T_2 = W_2 + W_3 + W_4$
$T_3 = W_3 + W_4 + W_5$
$T_4 = W_4 + W_5 + W_6$
$T_5 = W_5 + W_6 + W_7$.

	W_1	W_2	W_3	W_4	W_5	W_6	W_7
T_1	Ø	S	E				
T_2	⇒	S	E	G			
T_3		⇒	E	G	N		
T_4			⇒	G	N	O	
T_5				⇒	N	O	Ø

Questo procedimento è applicato sia alle parole delle (D) descrizioni delle varie categorie previste nel dizionario della classificazione considerata (*Lookup file*), sia alle parole delle (T) descrizioni fornite dal rispondente sulle quali si vuole far eseguire una ricerca (*search argument*) per poter effettuare la codifica.

1.4.2.2 Gli algoritmi implementati per la codifica in Blaise: "Trigram search"

L'intero processo di ricerca dei Trigram può essere suddiviso in tre fasi:
- Search process,
- Select process
- Display process.

Search process
Questa fase serve per identificare quali Trigram della stringa di input debbano essere considerati "significativi" e debbano quindi entrare nella seconda fase.

Il "Search Process" è costituito da due step: prima selezione e creazione di un set per il calcolo del Best Score.

- Prima selezione
  La ricerca avviene per Trigram. Blaise esegue un confronto tra ogni Trigram dell'array (T) e ogni Trigram di tutte le descrizioni (D) e registra una prima tavola delle corrispondenze trovate.
  A questo punto il sistema esegue una prima selezione. I Trigram (T) trovati vengono accettati solo se sono coerenti con la condizione-soglia calcolata in base a due costanti numeriche predefinite:

a) *TrigramThresHoldFrac* (0,20)
b) *TrigramTresHoldNum* (1.000)

La doppia condizione in base a cui i Trigram vengono ignorati e quindi non sottoposti ai successivi processi è che contemporaneamente:

1) tali trigrammi sono trovati in un numero di Record del Lookup file che risulta essere maggiore del prodotto tra il numero totale dei record del Lookup file e la *TrigramTresholdFrac*

NumeroRecordTrigramTrovati > *TrigramTresHoldFrac* × *Numero Record del Lookup file*
(NumeroRecordTrigamTrovati > 0,20 × *NumeroRecord del Lookup file)*

2) tali trigrammi sono trovati in un numero di Record maggiore del valore assegnato alla *TrigramTresholdNum*

NumeroRecordTrigramTrovati > *TrigramTresholdNum*
(NumeroRecordTrigamTrovati >1.000*)*

La determinazione dei valori assegnati a *TrigramTresHoldFrac* e a *TrigramTresHoldNum* discrimina i Lookup file in base alla dimensione. Per esempio, i valori delle due soglie discriminano i file ≤ 5.000 da quelli ≥ 5.000.

- Lookup file ≤ 5.000.
  Per Lookup file ≤ 5.000 Record, il cui 20 per cento (*TrigramTresHold Frac* × *NumeroRecord del Lookup file*) è necessariamente ≤ 1.000, la condizione discriminante, ai fini dell'esclusione di un Trigram, è data dalla *TrigramTresholdNum* = 1.000.
  Ad esempio, nel caso di un Lookup file di 3.500 Record, se un Trigram qualunque fosse trovato 701 volte, la prima condizione sarebbe soddisfatta.
  Infatti 701 > 700 (dove 3.500 × 20/100 = 700).
  Tuttavia, non sarebbe soddisfatta la seconda.
  Infatti 701 < 1.000.
  Perciò questo ipotetico Trigram non verrebbe ignorato (evidentemente per essere ignorato dovrebbe avere una frequenza di almeno 1.001 Record.).

- Lookup file > 5.000.
  Per Lookup file > 5.000, il cui 20 per cento è necessariamente > 1.000, la condizione discriminante, ai fini dell'esclusione di un Trigram, è data dalla *TrigramTresHoldFra*c × *NrRecLookup file*.
  Ad esempio, nel caso di un Lookup file di 15.000 Record, se un Trigram fosse trovato un numero di volte > 1.000 e dunque la seconda condizione venisse soddisfatta, tuttavia la prima verrebbe soddisfatta solo se il numero di volte in cui il Trigram è stato trovato, fosse non solo > 1.000, ma anche > 3.000 (dove 15.000 × 20/100 = 3.000).
  Perciò, se questo ipotetico Trigram fosse trovato 3.001 > 3.000 verrebbe ignorato.
  Mentre, un Trigram con frequenze ≤ 3.000, non potendo essere mai soddisfatta la prima condizione, verrebbe considerato e sottoposto ai successivi processi.

Tutto questo si può riassumere dicendo che, per Lookup file di dimensioni relativamente contenute (fino a 5.000 Record), un Trigram è considerato significativo per il corretto ritrovamento della descrizione anche quando abbia un'elevata frequenza (cioè fino a 1.000 record).
Mentre si può dire che, per Lookup file > 5.000, un Trigram anche di elevata frequenza (cioè > 1.000) è considerato significativo finchè questa non supera il 20 per cento della dimensione totale del Lookup file.

- Creazione di un set e calcolo del BestScore.

Per ciascun Trigram che abbia superato questo primo test viene creata una tabella con gli identificativi dei record del Lookup file che contengono il Trigram e con il loro corrispondente numero di Trigram. Le tabelle relative a ciascun Trigram vengono quindi riunite in un'unica tabella in cui a ciascun record del Lookup file, viene associato il numero di tutti i "Search Trigram" trovati (Record Score). A questo punto viene calcolata una graduatoria e individuato il *BestScore* (Record con il punteggio più alto. Corrisponde al massimo numero di Trigram trovati).

Select process

In questa fase Blaise calcola una ulteriore soglia (*TresHoldScore***)** che discrimina quali dei record del Lookup file, che hanno superato il primo test, debbano essere proposti in visualizzazione.
Il valore di questo filtro non è una costante impostata una volta per tutte, ma viene definito a seguito di ciascun Search process, tenendo conto di alcune costanti numeriche e di due valori empirici.

Costanti numeriche:

TrigramCutOffBestScore = 0.76 (probabilità del miglior punteggio empirico)
TrigramCutOffMaxScore = 0.51 (probabilità del massimo punteggio)
TrigramScoreTreshold = 2 (numero minimo di Trigram che rappresenta la soglia per il Display; questo valore può essere modificato nella Modelibrary)

Valori empirici:

BestScore (massimo punteggio corrispondente al record del Lookup file con il massimo numero di trigrammi trovati)
NrOfTrigram (numero di trigrammi della stringa di ricerca)

Il valore assegnato da Blaise al **TresHoldScore** corrisponderà al **massimo** tra i seguenti valori:

- **TrigramScoreTreshold**
- **Valore minimo** tra:

$$TrigramCufOffBestScore \times BestScore$$
$$TrigramCufOffMaxScore \times NrOfTrigram$$

Analizziamo le diverse possibilità che possono verificarsi:
Se

$$TrigramCufOffBestScore \times BestScore > TrigramCufOffMaxScore \times NrOfTrigram$$

Allora

$TrigramCufOffMaxScore \times NrOfTrigram$ viene confrontato con *TrigramScoreTreshold*.

Se

$$TrigramCufOffMaxScore \times NrOfTrigram > TrigramScoreTreshold$$

Allora

$$\textbf{TresHoldScore} = TrigramCutOffMaxScore \times NrOfTrigram$$

Altrimenti

$$\textbf{TresHoldScore} = TrigramScoreTresHold.$$

Se invece

$$TrigramCufOffBestScore \times BestScore < TrigramCufOffMaxScore \times NrOfTrigram$$

Allora

$TrigramCufOffBestScore \times BestScore$ viene confrontato con $TrigramScoreTreshold$.

Se

$$TrigramCufOffBestScore \times BestScore > TrigramScoreTreshold$$

Allora

$$\textbf{TresHoldScore} = TrigramCufOffBestScore \times BestScore$$

Altrimenti

$$\textbf{TresHoldScore} = TrigramScoreTreshold$$

Per chiarire quali siano gli effetti di questa logica sulle diverse tipologie di applicazioni di codifica che possono essere implementate, si riportano alcuni esempi.

Esempio A

Prendiamo il caso più semplice costituito da una qualunque stringa di tre caratteri. Diamo per scontato che, superato il primo test, i Trigram trovati siano stati associati ad un set di record ordinati in una graduatoria sulla base del loro punteggio (Record Score dato dalla somma di tutti i Trigram trovati in ciascuno Record.). Ricordiamoci che il sistema ha già calcolato il punteggio migliore BestScore e quello massimo MaxScore (MaxScore = NrOfTrigram).
Immaginiamo adesso che la ricerca della stringa (MaxScore = 3) abbia ottenuto BestScore = 2.

Passiamo a calcolare il filtro:

1° confronto: scelta del minimo $(0{,}76 \times 2) = \mathbf{1{,}52} < (0{,}51 \times 3) = 1{,}53$

2° confronto: scelta del massimo $\mathbf{2} > 1{,}52$

Il filtro selezionato per il Display Process è 2 (*TrigramScoreTreshold*). In questo caso la soglia predefinita non è modificata.

Se avessimo avuto un Exact match (Best Score=MaxScore) il calcolo del filtro sarebbe stato il seguente:

1° confronto: scelta del minimo $(0{,}76 \times 3) = 2{,}28 > (0{,}51 \times 3) = \mathbf{1{,}53}$

2° confronto: scelta del massimo $\mathbf{2} > 1{,}53$.

Il filtro selezionato per il Display Process è sempre 2. Anche in questo caso la soglia predefinita non è modificata.
Il filtro selezionato è in entrambi i casi uguale alla soglia 2. Come si vede dagli esempi, questo modo di procedere evita che il filtro possa essere inferiore alla soglia predefinita.

Esempio B

Prendiamo un caso più complesso costituito da una qualunque stringa di 10 caratteri. Come nel caso precedente, assumiamo che, superato il primo test, i Trigram trovati siano stati associati ad un set di Record ordinati in una graduatoria sulla base del loro Record Score .
Dato MaxScore = 10, immaginiamo che la stringa di ricerca abbia ottenuto BestScore = 4.

Passiamo a calcolare il filtro:
1° confronto: scelta del minimo $(0{,}76 \times 4) = \mathbf{3{,}04} < (0{,}51 \times 10) = 5{,}1$
2° confronto: scelta del massimo **3,04** > 2.
Il filtro selezionato per il Display Process è 3,04 ($TrigramCufOffBestScore \times BestScore$). Il filtro calcolato ha modificato la soglia predefinita.

Se avessimo avuto un Exact match (Best Score=MaxScore) il calcolo del filtro sarebbe stato il seguente:
1° confronto: scelta del minimo $(0{,}76 \times 10) = 7{,}6 > (0{,}51 \times 10) = \mathbf{5{,}1}$
2° confronto: scelta del massimo **5,1** > 2.
Il filtro selezionato per il Display Process è 5,1 ($TrigramCufOffMaxScore \times NrOfTrigram$). Il filtro calcolato ha modificato la soglia predefinita.

Il filtro selezionato è in entrambi gli ultimi due casi superiore alla soglia predefinita (2). Come si vede dall'esempio, questo modo di procedere evita che il filtro possa essere superiore al $TrigramCufOffMaxScore \times NrOfTrigram$, il che equivale a dire che la probabilità associata al miglior risultato possibile possa essere superiore a quella del 51 per cento associata al massimo risultato. Questo permette di selezionare i record da visualizzare a Display sulla base di un filtro più restrittivo della soglia predefinita, ma comunque meno restrittivo di quello che potrebbe derivare da Best Score prossimi a Max Score.
In sintesi nel caso di stringhe costituite da un elevato numero di Trigram, quanto più il Best Score si avvicina al MaxScore (ci si avvicina ad un Exact Match), quanto più tende ad innalzarsi la soglia (**TresHoldScore)** che discrimina quali record del Lookup file debbano essere proposti in visualizzazione.

Display process
In questa fase, viene definito l'ordine di visualizzazione dei Record selezionati.
I Record del set finale, selezionato in base al filtro calcolato dal sistema (TresHoldScore), sono mostrati in un ordine che risulta dall'effetto dell'interazione di due criteri: il punteggio considerato in ordine discendente (RecordScore) e la lunghezza della descrizione considerata in ordine ascendente (Length).

RecordScore. Ogni Record è ordinato in ordine discendente in base al proprio punteggio. I record con il punteggio più alto occupano i primi posti. Nel caso di match esatto (i Trigram cercati sono trovati tutti in una stesso Record), questo occupa il primo posto (compatibilmente con il criterio Lenght) .
Lenght della descrizione. Ogni Record è ordinato in ordine ascendente. Le descrizione più brevi occupano i primi posti.

L’effetto dell’interazione di questi due criteri crea un set in cui ai primi posti figurano i Record con le descrizioni più brevi e con il maggior numero di Trigram trovati. In questo modo si cerca di approssimare un match il più preciso possibile. Infatti tanto più il maggior numero di Trigram trovati è concentrato in un descrizione breve, tanto più si riduce la probabilità di descrizioni diverse da quella cercata.
È anche possibile che questo processo dia luogo a qualche incongruenza. Ad esempio, si potrebbe verificare un match esatto quando il numero di Trigram, che danno luogo a un match tra D e T, è lo stesso, pur non abbinandosi tutti i Trigram di T: ciò è possibile perché uno o più Trigram di T possono essere ritrovati in D più di una volta. Oppure, il fatto che ciascun Trigram sia cercato indipendentemente l’uno dall’altro, senza mantenere un riferimento alla stringa di ricerca, potrebbe dare luogo a match con alcuni particolari sotto insiemi di senso compiuto diverso da quello della parola alla quale appartengono (si potrebbero creare delle “parole potenziali” come avviene ad esempio, per la parola “decoratore” nella quale si trovano contenute “eco/oratore/ore”). Questo per dire che un criterio di ricerca dei trigrammi puramente quantitativo e combinatorio, in generale potrebbe comportare qualche bizzarro effetto che, tuttavia, nel caso di classificazioni estremamente specialistiche, risulta molto contenuto dalla omogeneità linguistica di questo tipo di testi.

2 Actr v3: *Automated Coding by Text Recognition*

2.1 Introduzione ad Actr

Come illustrato nella figura 1, Actr è un **sistema** costituito da una serie di componenti (dati, programmi, utility), la cui funzione fondamentale è quella di assegnare codici (secondo classificazioni predefinite) a descrizioni. Ciò viene realizzato attraverso il *"match"* tra i testi da dare in input al sistema, provenienti dai questionari delle rilevazioni statistiche, e le descrizioni standardizzate (sottoposte al *parsing*) presenti nel database di Actr.
In quanto generalizzato, può essere utilizzato ad esempio per codificare la professione, l'attività economica, le cause di morte, il titolo di studio; sarà l'utente di Actr a fornire al sistema lo schema di codifica, effettuando il *LOADING* di uno o più database con codici e descrizioni ed ottimizzando l'ambiente di codifica attraverso la messa a punto sia dei *file di parsing,,* sia dei parametri di sistema (soglie).
I database di Actr contengono coppie di codici e descrizioni ricavate dai manuali ufficiali delle classificazioni di riferimento e da precedenti indagini nelle quali è stata rilevata la variabile sotto osservazione. Queste descrizioni, prima di essere caricate (con l*'utility LOAD*) nel database potevano contenere, oltre a parole significative, blank di troppo, articoli, congiunzioni, preposizioni, parole ininfluenti, eccetera. Parte del processo di *LOADING* è proprio l'effettuazione della standardizzazione (*parsing*) di queste descrizioni in modo da ridurle ai loro componenti principali, secondo una "*strategia di parsing*" definita in funzione della lingua utilizzata e del contesto applicativo.
Quando Actr codifica, la descrizione in *input* è sottoposta alla stessa *strategia di parsing* utilizzata in fase di *LOADING*. La risultante descrizione standardizzata (e non quella originale) viene quindi usata per la ricerca nel database.
Come può dedursi, quindi, la gestione della strategia di *PARSING* è il punto cruciale del sistema, che più di tutti influisce sulle sue *performance*.
Per questo motivo, in questo manuale, si è deciso di introdurre per prima la problematica inerente il *parsing* e successivamente la descrizione delle altre componenti del sistema.

2.1.1 Computing Platforms

Actr v3 supporta Intel-based Windows NT 3.5 e Windows 95 (e successive versioni), le varianti di Unix Solaris, HP/UX e IRIX su SUN, rispettivamente con hardware HP e SGI.
Dal momento che il cuore di Actr è un portabile in *"C"* (esclusa la Graphical User Interface), il numero di piattaforme supportate può essere esteso.
La versione disponibile attualmente presso l'Istat è quella in ambiente Windows.

2.1.2 Interfaccia utente

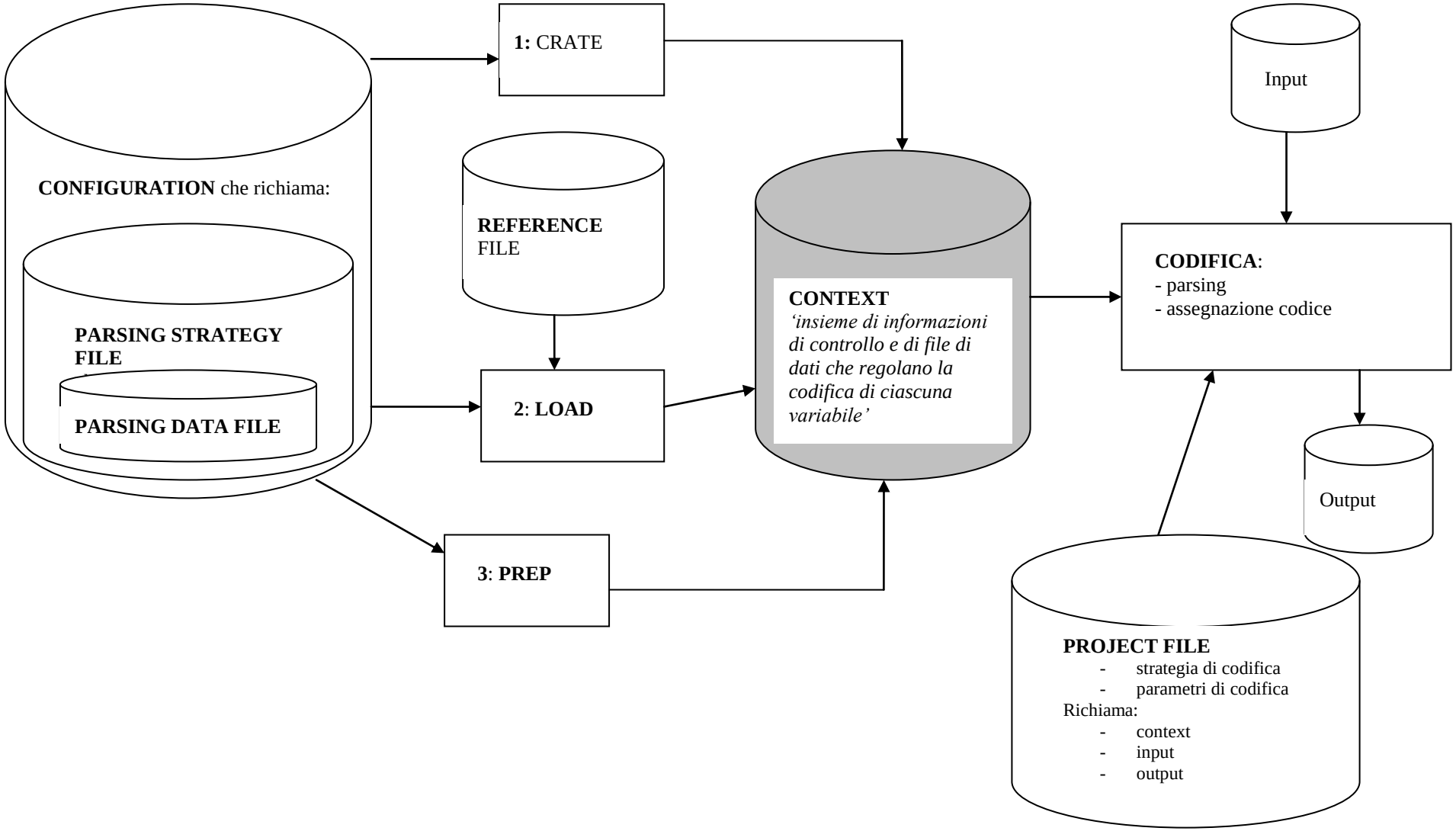
La prima versione di Actr v3 funzionava esclusivamente con una Command Line Interface (CLI), disponibile sia dal prompt di sistema che da una *"Shell"*. Entrambe utilizzano la stessa sintassi, tuttavia, mentre la CLI lavora su un singolo task alla volta e rilascia tutti i file ed i buffer di memoria a conclusione di ciascun task, la *"Shell"* consente all'utente di eseguire una serie di task senza dover lanciare Actr ogni volta.
Più recentemente, è stata resa disponibile una Graphical User Inteface (GUI) (confrontare paragrafo 2.4.5) che gira in ambiente Microsoft Windows e realizza molte (ma non tutte) le funzioni disponibili con la CLI ed alcune altre non disponibili con la CLI.

2.1.3 Programmare con le API

Actr v3 può anche essere richiamata direttamente tramite programmi *"C"* scritti dall'utente che richiamano le funzioni di Actr API (Application Programming Interface).
Mentre la CLI garantisce l'accesso in batch alle principali funzioni Actr , con le API si può ottenere una maggiore flessibilità (come per esempio l'utilizzo simultaneo di più contesti), oppure possono essere realizzate particolari elaborazioni richieste da specifiche applicazioni.

Per la documentazione sulle API si rimanda alla manualistica su Actr prodotta da Statistics Canada (disponibile in linea).

Figura 1 – Schema tecnico di Actr



2.2 Guida al Parsing

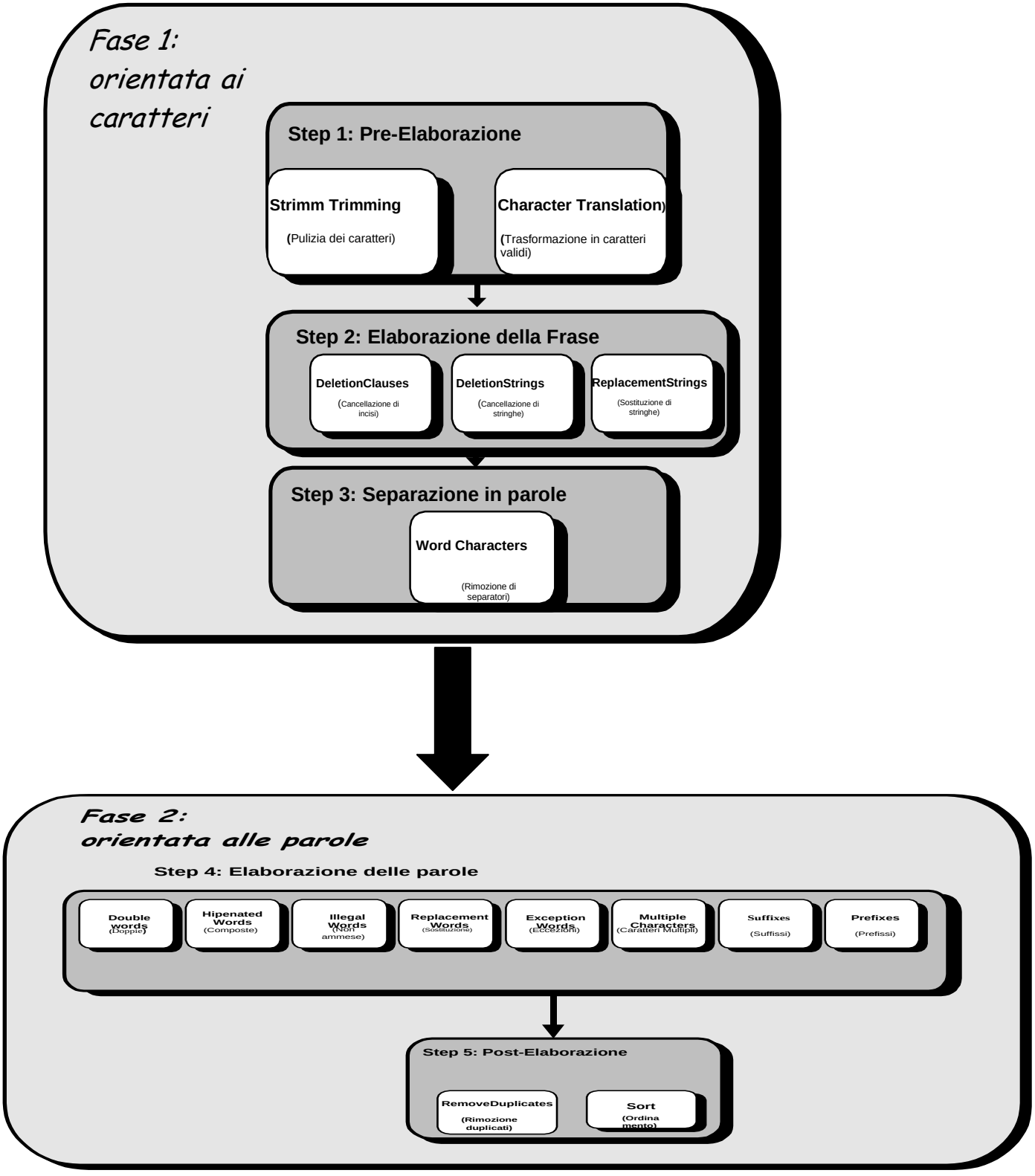
Il *PARSING* è parte integrante del processo di codifica di Actr; attraverso il *parsing* si riduce il testo ad una forma *"standardizzata"*.
Il risultato del *PARSING* deve essere tale che due descrizioni contenutisticamente uguali siano considerate identiche nella loro rappresentazione per Actr, indipendentemente dalle originarie differenze grammaticali o sintattiche. Per esempio, le due descrizioni: «`fabbricazione scarpe`» e «`produzione di calzature`» potranno essere uniformate, a seguito del *PARSING*, a «`produzion calzatur`».
La **strategia di parsing** è flessibile e facile da manutenere ed è definita in funzione di ciascun progetto di codifica. Ci sono tre meccanismi per intervenire sulla strategia:

- cambiamento dei tipi di processi di *PARSING* da utilizzare nel progetto,
- cambiamento dell'ordine di esecuzione dei singoli processi,
- cambiamento dei dati utilizzati nei processi di *PARSING* (contenuto dei file di *PARSING*).

2.3 Il meccanismo del Parsing

La strategia di *PARSING* definisce il modo in cui il testo di input è ricondotto ad una rappresentazione standardizzata. La strategia si articola in *fasi*, *step di parsing*, *processi* e *dati*.
Una rappresentazione grafica delle relazioni tra questi componenti è riportata nella figura 2.
La realizzazione del *PARSING* di Actr è logicamente suddivisa in due *fasi* principali. La differenza tra queste due *fasi* risiede nel modo in cui i testi sono trattati.
Nella "*fase orientata ai caratteri"*, tutte le manipolazioni del testo vengono effettuate " carattere per carattere". Le parole presenti nel testo non vengono riconosciute in quanto tali fino alla seconda fase del processo: la fase "*orientata alle parole"*.
Nell'ambito di queste due *fasi*, ci sono cinque *step* di *parsing*. Nell'ambito di ognuno di questi *step*, l'utente ha il controllo in merito a quale *processo* debba essere eseguito e come debba essere eseguito. Ciò è realizzato attraverso l'utilizzo dei *dati di parsing*. Modificando il contenuto di questi ultimi, viene alterata la *performance* dei processi corrispondenti.
Inoltre, nell'ambito degli *step* di *"elaborazione della frase"* ed *"elaborazione delle parole"* è possibile esercitare un ulteriore controllo, decidendo l'ordine con cui ogni singolo *processo* sarà eseguito.
Come avviene per qualsiasi procedura complessa, il meccanismo del *parsing* è più comprensibile analizzandone individualmente ogni singolo componente.
Nei prossimi paragrafi si entrerà in dettaglio nella presentazione di ogni componente, approssimativamente nello stesso ordine in cui sono rappresentati nella figura 2, alla pagina seguente.

Figura 2 – Fasi, step e processi del Parsing



2.3.1 Fase 1: orientata ai caratteri

Questa fase è composta da 3 step:

- *"Pre-Elaborazione"*
- *"Elaborazione della Frase"*
- *"Separazione in parole"*.

In questa fase il testo è elaborato come una stringa continua di caratteri. Nonostante si possa pensare che un testo sia costituito da parole, spazi e punteggiatura, nessuno di questi elementi, è fino a questa fase riconosciuto in quanto tale da Actr, e tanto meno preso in considerazione. Ne consegue che questa fase non esamina l'aspetto grammaticale, l'ordine delle parole, né nessun altro elemento inerente il contenuto informativo.
Il principale obiettivo di questa fase è di facilitare il riconoscimento di particolari sequenze di caratteri, così come sono stati rilevati, al fine di prevenire eventuali perdite di informazione, che potrebbero verificarsi quando il testo viene successivamente considerato in funzione delle parole che lo costituiscono.
Il passo finale di questa fase è l'individuazione delle singole parole. Ciò potrebbe indurre a pensare che questa fase non sia effettivamente *"orientata ai caratteri"*, ma individuare l'insieme di caratteri che formano ogni parola, quelli che rappresentano la punteggiatura, i separatori, eccetera, dimostra che in realtà si tratta effettivamente di una fase orientata ai caratteri.
Come tutti gli altri processi gestiti da Actr, anche quelli appartenenti a questa fase sono controllati dai dati con i quali l'utente li alimenta.
Le sezioni che seguono descrivono ciascuno dei tre step che costituiscono questa fase.

2.3.1.1 Fase 1. Step1: Pre-Elaborazione

Questo step è composto di 2 *processi* (funzioni), che sono eseguiti sempre nello stesso ordine così come appaiono in figura 2:

- *"String Trimming"*→ Pulizia dei caratteri,
- *"Character Translation"* → Traduzione dei caratteri

String Trimming

Lo *"String Trimming"* è un processo che standardizza il testo eliminando i caratteri estranei, quali spazi multipli, caratteri di tabulazione, di accapo, eccetera.
È specificato nella *strategia di parsing* (confrontare paragrafo 2.4.4.3), con *Autotrim yes/no.*
Possono essere anche specificate liste di tali caratteri rispetto ai quali bisogna rimuovere **tutti** i blank sulla destra (*TrimRight),* oppure sulla sinistra (*TrimLeft).*
Se si decide di prevedere tale processo nella *strategia di parsing,* questo dovrà precedere tutti gli altri processi.

Esempio:

- «COLTIVAZIONE DI CEREALI (ESCLUSO IL RISO)»,
- «COLTIVAZIONE DI CEREALI (ESCLUSO IL RISO)».

La differenza tra le due stringhe è lo spazio dopo la parentesi.

Vista la presenza di questo spazio, il primo testo non sarebbe elaborato correttamente da alcun processo nel quale ci si aspettava di trovare la stringa «(ESCLUSO».
Se invece fosse stato specificato e nella lista *TrimRight* il carattere « **(** », lo spazio tra « (» e «ESCLUSO» sarebbe stato rimosso prima del successivo processo che trattasse la stringa «(ESCLUSO»; in questo modo i due testi dell'esempio subirebbero un trattamento uniforme.

Character Translation

Questa funzione usa una lista di caratteri validi di riferimento e la relativa trasformazione per tradurre i caratteri dell'intera descrizione di input.
Tale lista è specificata dall'utente nel file dati "*Word Characters"* → **WChr.GPD** (confrontare paragrafo 2.3.1.3).
Con questo processo si possono, per esempio, tradurre tutti i caratteri dal minuscolo al maiuscolo. In tal caso, si specifica «A» come carattere di riferimento in cui tradurre «a», «B» per «b», eccetera; la stessa cosa può essere effettuata per i caratteri accentati (à → A).
In questo modo, i caratteri presenti nella lista saranno tradotti, mentre gli altri resteranno inalterati, fino al successivo step di «Separazione in parole» (il modo in cui tali caratteri saranno utilizzati per separare le parole è descritto nel paragrafo 2.3.1.3).
Bisogna sempre tenere a mente che ciò che viene previsto in questa fase del *parsing* ha un impatto su tutti i successivi processi. Per esempio, se in questo processo tutte le minuscole sono state tradotte in maiuscole, tutti i dati nei *file di parsing* dei processi successivi devono essere in maiuscolo.

2.3.1.2 Fase 1. Step2: Elaborazione della frase

Questo step è composto di 3 *processi* (funzioni), che sono tutti opzionali e, se specificati nella *strategia* (confrontare paragrafo 2.4.4.3), possono essere eseguiti in qualsiasi ordine:

- *"Deletion Clauses"* → Cancellazione di incisi,
- *"Deletion Strings"* → Cancellazione di stringhe,
- *"Replacement Strings"* → Sostituzione di stringhe.

Si noti che se è stato previsto lo *"String Trimming"* nella *strategia di parsing*, le occorrenze multiple di spazi saranno rimosse a seguito di ognuno di questi processi. Ciò assicura che la sequenza di caratteri rimanga in formato standard, nonostante gli effetti di sostituzione o rimozione di parti di testo.

Deletion Clauses

È un metodo per delimitare una frase che dovrebbe essere rimossa dal testo di input. La frase viene individuata fornendo una stringa di inizio e una di fine, così tutto ciò che è compreso tra le due citate stringhe viene rimosso.
Tale lista è specificata dall'utente nel file dati *"Deletion Clauses"* → **DCls.GPD**

Esempio: si consideri una sintassi di *"Deletion Clauses"* definita con una stringa di inizio «(ESCLUSO» ed una stringa di fine «)».
Includendo ciò nella *strategia di parsing*, le due stringhe:

«PROGRAMMAZIONE DI SOFTWARE (ESCLUSO IMPIEGATI)»,

«PROGRAMMAZIONE DI SOFTWARE (ESCLUSO LAVORATORI AUTONOMI)»,

sarebbero uniformate dal *parsing* alla stessa descrizione → «PROGRAMMAZIONE DI SOFTWARE».

Deletion Strings

I dati elencati nel processo di *"Deletion Strings"* vengono rimossi, in qualsiasi posizione della descrizione in input essi si trovino. Questo processo viene abitualmente utilizzato per rimuovere caratteri che altrimenti trattati come delimitatori (apostrofi, parentesi).
Tale lista è specificata dall'utente nel file dati *"Deletion Strings"* → **DStr.GPD**

Un esempio tipico è per la lingua inglese l'apostrofo; includendo questo carattere nelle *"Deletion Strings"*

→«ELECTRICIAN'S APPRENTICE»,

→«ELECTRICIANS APPRENTICE»

Sarebbero uniformate.

Tale processo viene tuttavia spesso utilizzato anche per rimuovere stringhe di caratteri ritenute ininfluenti ai fini della codifica. Nonostante che la *"Deletion Strings"* sia molto utile, bisogna fare molta attenzione nel suo utilizzo. Infatti, se l'intento fosse quello di rimuovere tutte le occorrenze della parola "DEL", non soltanto si otterrebbe la rimozione della citata preposizione articolata, ma di tutte le occorrenze di questa stringa rilevate nelle descrizioni. Quindi,

→«DELTA» diverrebbe TA»,

→«ADELE» diverrebbe «AE».

Non bisogna infatti dimenticare che, in questa fase del *parsing*, si stanno ancora trattando esclusivamente sequenze di caratteri (stringhe) e non ancora parole.

Un altro problema che si potrebbe incontrare in questa fase è quello del susseguirsi di molteplici caratteri definiti come non validi (esempio blank).
Per esempio, se si volesse rimuovere la stringa «WEEK END», ma ci si trovasse in input una descrizione contenente «WEEK END», quest'ultima non sarebbe rimossa a causa della presenza dei due blank.
Come già detto in precedenza, però, qualora sia stato previsto lo *"String Trimming"* (confrontare paragrafo 2.3.1.1) nella *strategia di parsing*, la presenza di molteplici blank non costituirebbe più un problema.

N.B. Il testo dopo la eliminazione dei caratteri, sarà compresso: al posto dei caratteri eliminati non ci saranno blank, infatti la lunghezza della stringa sarà ridotta.

Replacement Strings

Questo processo è utilizzato soprattutto per **abbreviazioni standard**.
Questa soluzione è consigliata dal momento che spesso queste abbreviazioni contengono caratteri che sarebbero altrimenti trattati come separatori e, se ciò succedesse, probabilmente si procurerebbe una perdita di informazioni.
Si pensi per esempio alla stringa «`T.V.`», da sostituirsi con «`TELEVISIONE`».
Se tale sostituzione non fosse fatta in questo processo ed il carattere « `.` » fosse stato giustamente considerato un delimitatore (quindi, escluso dalla lista dei caratteri validi per costituire parole), «`T.V`.» verrebbe spezzato in due elementi singoli e non più considerato nel suo insieme.

È opportuno inoltre fare un'altra precisazione: si consideri la descrizione: «`T.V.STORE`».
Se nelle *"Replacement Strings"* «`T.V.`» fosse semplicemente sostituita da «`TELEVISIONE`», la descrizione trasformata sarebbe: «`TELEVISIONESTORE`», perdendo la possibilità di suddividere le due parole.
Bisognerà quindi prevedere una sostituzione che mantenga tale separatore, per esempio: «`#TELEVISIONE#`», in modo che la descrizione trasformata diverrà: «`#TELEVISIONE#STORE`». Anche così sembrerebbe essersi perso il confine tra una parola e l'altra; tuttavia se il carattere « **`#`** » fosse stato giustamente considerato un delimitatore (quindi, escluso dalla lista dei caratteri validi per costituire parole), la stringa «`#TELEVISIONE#STORE`» verrebbe spezzata in due parole singole, ossia «`TELEVISIONE`» «`STORE`».
È evidente quindi che **Replacement Strings**, essendo molto potente, debba essere usato con estrema cautela.
Tale lista è specificata dall'utente nel file dati *"Replacement Strings"* → **RStr.GPD**
Un'altra finalità per cui è utilizzabile questo processo è quello di rimuovere dalla descrizione sequenze di parole ininfluenti ai fini dell'attribuzione del codice.
Prevedendo per esempio di sostituire «`PER TUTTI GLI USI`» con blank, le due stringhe

«PRODUZIONE DI UTENSILI»,
«PRODUZIONE DI UTENSILI PER TUTTI GLI USI»,

sarebbero uniformate.
Si ricorda comunque che la trasformazione in blank delle stringhe può essere anche effettuata nelle DStr.
Infine, in molti step (a livello sia di elaborazione della frase che delle parole) deve essere presa una decisione in merito al fatto se sia più opportuno espandere o ridurre il testo.

Per meglio spiegare questo problema, si pensi alle due stringhe:
→ «ISTITUTO NAZIONALE DI STATISTICA»,
→ «ISTAT».
Ridurre tutto alla sigla comporterebbe certamente tre vantaggi:

- a livello di attribuzione dei pesi, quindi del punteggio finale, la sigla verrebbe pesata come una singola parola;
- dal momento che «ISTAT» è una singola, breve descrizione, non ci si dovrebbe preoccupare di eventuali altri processi di *parsing* su questa stessa;
- trattandosi di una singola parola, costituita da poche lettere, il *parsing* sarà eseguito più velocemente.

Tutti questi vantaggi possono essere parimenti letti come svantaggi della descrizione estesa.
Tuttavia quest'ultima produrrebbe un vantaggio che non si otterrebbe con la sigla: ci sarebbero un maggior numero di match parziali (con le altre descrizioni che contengono una delle parole costituenti «ISTITUTO NAZIONALE DI STATISTICA»). Questo è particolarmente vantaggioso in caso di eventuali errori di ortografia in una delle parole della versione estesa; infatti la presenza di un errore non consentirebbe la sostituzione con la sigla.
Quindi, laddove, la versione estesa è costituita da più di due parole (e perciò è maggiore il rischio di errori) è preferibile ricondurre la sigla alla versione estesa, piuttosto che fare il contrario.
In sintesi, nella decisione finale bisogna tenere conto del fatto che il processo orientato alle parole è senza dubbio più semplice e veloce di quello orientato ai caratteri, quindi, laddove possibile, da preferire.

2.3.1.3 Fase 1. Step3: Separazione in parole

Actr definisce una parola come una qualsiasi sequenza di caratteri contigui, che siano tutti previsti in una lista di caratteri validi.
Tale lista è specificata dall'utente nel file dati *"Word Characters"* → **WChr.GPD.**
I caratteri non inclusi in tale lista sono considerati come delimitatori delle parole e non saranno quindi oggetto di ulteriori elaborazioni da parte del *parsing*.

N.B. Il blank non può far parte della lista dei caratteri validi.
(Il trattino sarà successivamente spiegato nel paragrafo 2.3.2.1 "Hyphenated Words")

Word Characters

Tipicamente l'insieme di caratteri validi contempla tutte le lettere dell'alfabeto e tutti i caratteri numerici.
I caratteri non contenuti in questo insieme sono considerati separatori e quindi rimossi (sostituiti da blank), consentendo la separazione del testo in parole.

Per esempio: la descrizione «PESCATORE / VENDITA PESCE» risulterà composta da tre parole, poiché il carattere « / » è considerato un carattere delimitatore sostituito da blank; quindi la descrizione sarà trasformata in «PESCATORE VENDITA PESCE».

Il file dati *"Word Characters"*» → **WChr.GPD** è l'unico *file dati* del *parsing* che **non** può essere mai vuoto.
È infatti necessario perché oggetto del *parsing* è definire le parole. Se non vengono specificati i caratteri che le costituiscono, la definizione delle parole è impossibile.

2.3.2 Fase 2: Orientata alle parole

Dopo che la descrizione è stata suddivisa nelle parole che la costituiscono, Actr procede all'analisi orientata alle parole.
Ogni parola viene esaminata da ciascun processo del *parsing* previsto nella strategia, secondo l'ordine definito dall'utente.
Il *parsing* agisce sulla descrizione in input finché ogni parola è stata esaminata oppure addirittura finché non rimane alcuna parola della descrizione in input; infatti se tutte le parole del testo di input fossero sostituite da blank, l'output del *parsing* sarebbe blank.

Come evidenziato nella figura 2, questa fase è costituita da due step*:*
- *"Elaborazione delle parole"* → Word processing step,
- *"Post-elaborazione"*→ Post-process.

L'ordine degli step non può essere cambiato, ma l'ordine dei processi all'interno degli step e i dati utilizzati in ciascun processo sono modificati dall'utente.
È importante ricordare che tutto il *parsing* è regolato dalla *strategia di parsing* (confrontare paragrafo 2.4.4.3)
I dati del *parsing* possono essere o quelli forniti di default da Actr (un set molto piccolo con cui la standardizzazione è minimizzata), eventualmente modificabili dall'utente, oppure interamente creati da quest'ultimo.
Lo stesso processo di *"Word Characters"* non può essere soppresso, ma i dati possono essere modificati per riflettere la *strategia di parsing* definita dall'utente.
I successivi paragrafi descrivono ciascuno i relativi processi.

2.3.2.1 Fase 2. Step1: Elaborazione delle parole

Questo step prevede 8 processi, nessuno dei quali è obbligatorio e il cui ordine può essere definito dall'utente:

- *"Double Words"* → Parole doppie,
- *"Hyphenated Words"* → Parole Composte,
- *"Illegal Words"* → Parole non ammesse,
- *"Replacement Words"* → Sostituzione di parole,
- **"***Exception Words***"** → Parole eccezione,
- *"Multiple Characters"* → Caratteri multipli,
- *"Suffixes"* → Suffissi,
- *"Prefixes"*→ Prefissi.

Tuttavia, l'ordine dei processi non è casuale: la sequenza scelta deve garantire la coerenza logica della successione degli effetti delle trasformazioni realizzate. L'unico processo che deve sempre occupare una posizione precisa è quello definito *"Exception Words"*. Infatti, questa funzione, impedendo le trasformazioni relative alle lettere multiple, ai suffissi e ai prefissi, richiede di essere indicata prima che queste siano già avvenute.

Double Words

Questo processo costringe Actr a considerare non soltanto le occorrenze di gruppi di due parole, ma anche il loro ordine. Questo può essere utile per:

- preservare l'ordine delle coppie di parole,
- trattare in maniera uniforme parole composte,
- risolvere eventuali abbreviazioni di per sé non univoche, se non in funzione delle parole con cui sono abbinate.

A chiarimento di quanto detto, si pensi ai seguenti esempi:

→ la descrizione **«SUB ORDINE»**, costituita da due parole, non sarà riconosciuta equivalente a **«SUBORDINE»**, a meno che non venga sostituita con quest'ultima nel processo delle *"Double Words"*. Qualora poi il « – », non sia stato definito nel WChr come un carattere valido, anche **«SUB-ORDINE»** sarebbe trasformato in **«SUBORDINE»**, a seguito delle DWrd.

→ L'abbreviazione **«PROD»** non è univoca, in quanto può significare alternativamente **«PRODUZIONE»** oppure **«PRODOTTI»**. In questo processo si può decidere di ricondurla ad un significato oppure all'altro, a seconda delle parole con cui si abbina nel testo ed all'ordine in cui si trova rispetto ad esse:

«PROD COMMERCIO» → **«PRODUZIONE COMMERCIO»**
«COMMERCIO PROD» → **«COMMERCIO PRODOTTI»**

Tale lista è specificata dall'utente nel file dati *"Double Words"* → **DWrd.GPD**

Hyphenated Words

Questo processo consente il riconoscimento di parole e gruppi di parole che potrebbero essere scritte con un trattino « - » che ne suddivide le parti componenti; per esempio **«BABY-SITTER»** potrebbe essere equiparata a **«BABYSITTER».** Qualora questa trasformazione non sia stata prevista in questo processo, Actr considererebbe il trattino come un separatore e tratterebbe le parole singolarmente (si veda quanto detto nel paragrafo precedente a proposito di «SUB-ORDINE».
Una *"Hyphenated Words"* può essere costituita da un massimo di tre parole.
L'utilizzo delle HWrd è quindi necessario in due casi:

- qualora il carattere « - » sia definito come valido nel WChr (in caso contrario l'accorpamento di due parti componenti di una parola potrebbe essere realizzato nel processo Double Words)
- qualora la parola con trattino debba essere sostituita da tre parole.

Alcuni esempi di utilizzo di questo processo sono:

«VICE-AMBASCIATORE» → **«AMBASCIATORE»**
«BABY-SITTER» → **«BABYSITTER»**

Si sottolinea che, nel caso in cui si voglia mantenere la grafia della parola con il trattino, è indispensabile far sì che Actr lo riconosca come carattere valido ed inserirlo quindi negli *"Word Characters"*.
Per esempio, se si vuole mantenere il trattino nella parola «BABY-SITTER», è necessario non soltanto includere « – » negli *"Word Characters"*, ma anche inserire «BABY-SITTER» come sostituta di se stessa nelle *"Hyphenated Words"*.
Tale lista è specificata dall'utente nel file dati *"Hyphenated Words"* → **HWrd.GPD**

Illegal Words

La definizione di *"caratteri non ammessi"* consente di sottrarre dalle successive elaborazioni tutte le parole che li contengano.
Tale lista è specificata dall'utente nel file dati *"Illegal Words"* → **IWrd.GPD**

Per esempio, definire i caratteri numerici nelle *"Illegal Words"* comporta non soltanto la rimozione dei caratteri stessi, ma anche di tutte le parole che contengano numeri (es. «PROT012001A»).
È possibile anche definire come illegali esclusivamente particolari stringhe di caratteri. Per esempio, prevedendo nelle *"Illegal Words"* la sequenza di caratteri « 85B » saranno rimosse esclusivamente le stringhe di caratteri contenenti « 85B ».

Replacement Words

Questo è il processo che più di tutti realizza in senso vero e proprio la gestione dei sinonimi.
Consente infatti che una parola sia riconosciuta uguale ad un'altra oppure a due parole ai fini del matching.
Tale lista è specificata dall'utente nel file dati *"Replacement Words"* → **RWrd.GPD**

Esempi:

```
«IL»            →   «          »
«ANCHE»         →   «          »
«ANNESSO»       →   «          »
«ABBIGL»         →  «ABBIGLIAMENTO»
«INDUMENTO»          «ABBIGLIAMENTO»
«GONNA»              «ABBIGLIAMENTO»
«ABBILIAMENTO»  →   «ABBIGLIAMENTO».
«VITICOLTORE»        «COLTIVAZIONE VITE»
```

Come è evidente dagli esempi, questo processo è quindi utilizzabile per realizzare diverse funzioni:
- eliminare parole ininfluenti (sostituzione della parola con blank),
- gestire le abbreviazioni,
- ricondurre una parola ad un sinonimo che potrebbe essere costituito da una o due parole,
- gestire errori ortografici ricorrenti.

L'**abolizione delle parole ininfluenti** è tipicamente utilizzata per rimuovere articoli, preposizioni (semplici/articolate) e congiunzioni; inoltre può essere utile per eliminare quelle parole che non apportano un contributo significativo al testo da codificare (per esempio «ANNESSO»).
Relativamente alla **gestione delle abbreviazioni** è bene tenere in mente quanto segnalato nel paragrafo sulle *"Double Words"* a proposito delle abbreviazioni non univoche.

In generale, poi, in fase di **definizioni di sinonimi**, bisogna tenere presente tre possibilità:

- gestirli nelle *"Replacement Words"*,
- gestirli nelle *"Double Words"*,
- inserire una nuova occorrenza nel dizionario elaborabile.

La prima soluzione è la più radicale, adatta quando il sinonimo è sempre valido per tutte le classi della classificazione di riferimento e indipendentemente dalle altre parole della frase con le quali la parola in esame si può accompagnare.
La seconda soluzione è invece suggerita quando il sinonimo della parola in esame si esplicita a seconda del contesto in cui questa si inserisce.
Ad esempio, leggendo le due frasi:

«commercio mobile»,
«commercio ambulante»,

potrebbe venire la tentazione di stabilire nelle *"Replacement Words"* il sinonimo

«mobile» → «ambulante».

Tuttavia questa soluzione sarebbe errata, in quanto la parola «mobile» indica anche un oggetto di arredamento, sia nella lingua italiana, sia nello specifico contesto linguistico della classificazione delle Attività Economiche.
Casi come questo devono quindi essere risolti nelle *"Double Words"*, definendo il sinonimo:

«COMMERCIO MOBILE» → «COMMERCIO AMBULANTE».

Qualora invece il sinonimo sia peculiare del caso in esame e non possa in alcun modo essere generalizzato, è opportuno inserire una nuova occorrenza nel *"Reference file"* (confrontare paragrafo 2.4.4.6).
Infine, RWrd può essere utilizzato per **sanare gli errori ortografici più ricorrenti**, qualora non si sia in grado di anteporre alla fase di codifica automatica un passaggio di correzione ortografica basato sul dizionario di riferimento.

Exception Words

Prevedere una parola nelle *"Exception Words"* significa far sì che su questa non vengano eseguiti i processi finalizzati alla rimozione di caratteri multipli (*"Multiple Characters"*), suffissi (*"Suffixes"*) e prefissi (*"Prefixes"*).
Questo processo nella *strategia di parsing* (confrontare paragrafo 2.4.4.3*)* deve quindi essere eventualmente previsto prima dei processi citati nel precedente capoverso.
Questo processo deve essere utilizzato nei casi in cui rimuovere un suffisso/prefisso da una parola ne rende ambiguo il significato.
Tale lista è specificata dall'utente nel file dati *"Exception Words"* → **EXcp.GPD**

Esempio:

«BANCO» – «BANCA» – «BANCHI» – «BANCHE».

Qualora si preveda in Italiano di rimuovere i suffissi «A», «O», «HI» ed «HE», che tendenzialmente determinano il genere ed il numero, le quattro parole dell'esempio sarebbero uniformate alla parola **«BANC».**
Ciò comporterebbe il non riuscire più a distinguere i significati di

«BANCA» = «ISTITUTO DI CREDITO»,
«BANCO» = «TAVOLO», « SCRIVANIA».

È quindi opportuno prevedere nelle *"Exception Words"* almeno la parola che rappresenta uno dei due significati; quindi, per esempio:

«BANCA»,
«BANCHE»,

in modo che i suffissi non siano rimossi da queste due parole.
La stessa logica vale per i prefissi ed i caratteri multipli.

Multiple Characters

Nell'ambito di questo processo, viene esaminata in ogni singola parola la sequenza dei caratteri che la compongono in modo da individuare la presenza dei caratteri contigui (due o più) elencati nei *"Multiple Characters"*, così da ridurli ad un'unica occorrenza. Generalmente, se tale processo viene inserito nella *strategia di parsing* (confrontare paragrafo 2.4.4.3*)*, si adotta per tutti i caratteri dell'alfabeto.
Tale lista è specificata dall'utente nel file dati *"Multiple Characters"* → **MChr.GPD**

Per esempio, si avrebbe equivalenza per seguenti parole:

«ABIGLIAMENTO» → «ABBIGLIAMENTO»,
«PROGRAMATORE» → «PROGRAMMATORE».

Qualora non si voglia eseguire tale processo per alcune parole, è necessario che queste siano previste nelle *"Exception Words"*.

Questo processo consente di ridurre l'impatto degli errori ortografici sulla codifica automatica, ma è assolutamente sconsigliabile nel caso in cui le descrizioni da trattare siano costituite prevalentemente da nomi propri.

In tal caso, infatti, due nomi possono essere diversi proprio in funzione della presenza di **Double Characters**.

Suffixes

I suffissi più di frequente utilizzati nelle applicazioni di codifica automatica (soprattutto in Italiano) sono quelli che determinano numero e genere. In questo protocollo, sono definiti suffissi i gruppi vocalici e semivocalici posti alla fine delle parole e non i suffissi propri della lingua italiana.
Tale lista è specificata dall'utente nel file dati *"Suffixes"* → **Sufx.GPD**
A questo punto, le parole sono esaminate dal sistema da destra a sinistra in modo da individuare **prima i suffissi più lunghi** e poi quelli più brevi. Se viene trovato uno dei suffissi individuati, questo viene rimosso, **purché la parola restante sia lunga almeno quattro caratteri**.

Volendo chiarire con degli esempi, supponiamo di aver previsto i suffissi «A», «I», «IA» ed «HI».
Le seguenti parole saranno trasformate come segue:

«DRAG**HI**» → «DRAG»,
«MAGH**I**» → «MAGH»,
«AGHI» → «AGHI».

È importante sottolineare che, per ogni parola viene rimosso soltanto un suffisso (gruppo vocalico finale). Quindi, ipotizzando di aver previsto i già citati suffisi, la parola

«MANNA**IA**» → «MANNA»,

mentre l'ultima vocale «A» di «MANNA» non sarebbe rimossa. Se si volesse ovviare a questo problema, si dovrebbe prevedere anche il suffisso «AIA».
Qualora non si voglia eseguire tale processo per alcune parole, è necessario che queste siano previste nelle *"Exception Words"*.

Prefixes

I prefissi sono trattati come i suffissi, ad eccezione del fatto che la parola viene ridotta a partire dal suo inizio e che quindi viene esaminata da sinistra a destra.
Valgono quindi le stesse restrizioni citate per i suffissi.
Tale lista è specificata dall'utente nel file dati *"Prefixes"* → **Prfx.GPD**

2.3.2.2 Fase 2. Step2: Post-Elaborazione

Con lo step di *"Post-elaborazione"* si chiude il *parsing*.
Questo step prevede due processi:

- *"Rimozione dei duplicati"* → Remove Duplicates,
- *"Ordinamento delle parole"* → Sort

L'ordine di questi due processi (entrambi opzionali) non influisce sull'output del *parsing*. Tuttavia, se si decide di prevederli entrambi, sarebbe più efficiente prima rimuovere i duplicati e poi effettuare l'ordinamento, perché così la lista di parole da ordinare risulterebbe più breve.

Remove Duplicates

L'insieme di parole derivanti dai precedenti processi di *parsing* viene esaminato al fine di individuare eventuali parole che si ripetono più di una volta e di rimuovere i duplicati.
Si precisa che le parole che risultano duplicate in questa fase possono non essere state originariamente tali.

Si esamini per esempio la frase
`«COMMERCIO DI ABBIGLIAMENTO: GONNE E PANTALONI».`
Qualora nelle *"Replacement Words"* «GONNE» e «PANTALONI» fossero stati considerati sinonimi di «ABBIGLIAMENTO» e contemporaneamente fossero stati eliminati i suffissi «O», «I» e «IO», la frase originaria diverrebbe:
`«COMMERC ABBIGLIAMENT ABBIGLIAMENT ABBIGLIAMENT».`

Quindi, a seguito del processo di rimozione dei duplicati si otterrebbe:
`«COMMERC ABBIGLIAMENT».`

Sort

A questo punto le parole rimanenti a seguito dei processi già descritti vengono ordinate, secondo un ordine alfabetico ascendente.

2.3.3 Creare ed aggiornare la strategia di parsing

Finora sono stati descritti gli elementi componenti la *strategia di parsing* ed i meccanismi da usare per controllare il processo. Ulteriori dettagli su come modificare la strategia nell'ambito di ciascuna applicazione di codifica saranno forniti nel paragrafo 2.4.
L'esposizione sulla *strategia di parsing* è stata fatta *"step-by-step"* in modo da approfondire i dettagli, senza essere "sommersi" dall'intero processo.
È raccomandato lo stesso approccio in fase di sviluppo di un'applicazione. È infatti preferibile iniziare con un piccolo database ed una *strategia di parsing* minima, per poi ampliarla man mano che il database viene arricchito ed i primi risultati sono analizzati.
Infine si fa presente che, nella Guida all'utilizzo, saranno approfonditi i criteri standard da seguire nella costruzione di applicazioni di codifica automatica, nonché le relazioni tra un processo e l'altro ed i conseguenti accorgimenti da osservare nello sviluppare e manutenere il *parsing*.
Per chiarezza, si riassume quanto finora descritto nella seguente tavola, specificando per ciascun processo il nome o tipologia di file che lo gestisce.

Tavola 2 - Schema fasi-step-processi

Fase	Step	Processo	File
Fase 1 Orientata ai caratteri	**Step 1 Pre-elaborazione**	**1) Strim Trimming – pulizia dei caratteri:**	
		autotrim: cancellazione dei caratteri spuri, quali doppi blank, tabulatori, andata a capo, etc.	Context-name.GPS
		trimleft: cancellazione dei blank a sinistra del simbolo specificato, ad esempio ([!?\	Context-name.GPS
		trimright: cancellazione dei blank a destra del simbolo specificato, ad esempio ([!?\	Context-name.GPS
		2) Character Translation – traduzione dei caratteri: (nel file WChr.GPD sono riportati i caratteri validi e le relative trasformazioni – per es. da minuscolo a maiuscolo)	WChr.GPD
	Step 2 Elaborazione della frase	1) **Deletion Clauses – cancellazione degli incisi:** l'inciso è segnalato da un inizio/fine stringa, ad es. (esclus) (non)	DCls.GPD
		2) **Deletion Strings – cancellazione di stringhe:** elimina la stringa indicata, in qualsiasi posizione essa si trovi	DStr.GPD
		3) **Substitution Strings - sostituzione di stringhe:** sostituisce le stringhe, in qualsiasi posizione esse si trovino. Utile per standardizzare abbreviazioni, sigle, etc.; ad esempio: c.terzi → conto terzi	RStr.GPD
	Step 3 Separazione in parole	**Words Characters – separazione della stringa in parole:** (i blank ed i caratteri non presenti nel file WChr.GPD sono utilizzati come separatori per dividere la stringa in parole)	WChr.GPD
Fase 2 Orientata alle parole	**Step 1 Elaborazione delle parole**	1) **Double Words – parole doppie:** definisce coppie di parole in sequenza da sostituire con un'altra parola o coppia di parole	DWrd.GPD
		2) **Hyphenated Words – parole composte scritte con trattino:** definisce parole separate da trattino che debbono essere trasformate in un'unica parola o più parole.	HWrd.GPD
		3) **Illegal Words – parole non ammesse:** definisce stringhe che, se contenute in una parola, comportano l'esclusione	Iwrd.GPD

		della parola stessa	
		4) **Replacement Words – sostituzione di parole:** definizione di sinonimi e sostituzione di parole inutili con blank	RWrd.GPD
		5) **Exception Words – parole eccezione:** definizione di parole che <u>non</u> devono essere sottoposte all'eliminazione di prefissi/suffissi e caratteri multipli (da anteporre sempre ai processi MCHR, PRFX, SUFX)	Excp.GPD
		6) **Multiple Characters – caratteri multipli:** definizione di caratteri che, se doppi o tripli in sequenza all'interno di una parola, vengono ridotti ad uno solo	MChr.GPD
		7) **Suffixes – suffissi:** definizione di suffissi da eliminare a fine parola (solo se la parola troncata rimane almeno di 4 lettere)	SUfx.GPD
		8) **Prefixes – prefissi:** definizione di prefissi da eliminare a inizio parola (solo se la parola troncata rimane almeno di 4 lettere)	PRfx.GPD
	Step 2 **Post-Elaborazione**	1) **Remove Duplicates –** rimozione di parole doppie (RDUP)	Context-name.GPS
		2) **Sort** – ordinamento delle parole	Context-name.GPS

N.B. L'estensione **GPD** non è obbligatoria e può essere sostituita con **TXT**.

2.3.4 Formati dei file di parsing

Sono di seguito riportate le caratteristiche tecniche dei *file dati del parsing*, nell'ordine in cui sono citati nella tavola 2.

WCHR - Word Characters - (non può mai essere vuoto)

Col.	Dim.	Obbligatorio	Descrizione
1	1	Sì	Carattere da trasformare
2	1	Sì	Blank
3	1	Sì	Carattere in cui viene trasformato il carattere della colonna 1

DCLS - Deletion clauses

Qualunque frase in input che inizi e finisca con questi incisi viene cancellata dalla descrizione.

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Stringa iniziale, per esempio (ESCLUS
31-60	30	Sì	Stringa finale, per esempio)

DSTR - Deletion Strings

Col.	Dim.	Obbligatorio	Descrizione
1-50	50	Sì	Stringa che sarà cancellata

RSTR - Replacement Strings

Col.	Dim.	Obbligatorio	Descrizione
1 – 50	50	Sì	Stringa da sostituire
51-100	50	Sì	Stringa sostituta

DWRD - Double Words

Col.	Dim.	Obbligatorio	Descrizione
1 – 30	30	Sì	Prima parola da trasformare, per esempio PROD
31 – 60	30	Sì	Seconda parola, da trasformare per esempio VENDITA
61 – 90	30	Sì	Prima parola sostituita, per esempio PRODUZIONE
91-120	30	No	Seconda parola sostituita, per esempio COMMERCIO

HWRD - Hyphenated words

Col.	Dim.	Obbligatorio	Descrizione
1-60	60	Sì	Parola con trattino (hyphen) da sostituire, per esempio: BABY-SITTER
61 – 90	30	Sì	Prima parola sostituita, per esempio BABYSITTER
91-120	30	No	Seconda parola sostituita
121-150	30	No	Terza parola sostituita

IWRD - Illegal words

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Stringa da ricercare

RWRD - Replacement Words

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Parola da sostituire, per esempio: VITICOLTORE
31-60	30	Sì	Prima parola sostituita, per esempio: COLTIVAZIONE
61-90	30	No	Seconda parola sostituita, per esempio: VITE

EXCP - Exception Words

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Parola eccezione (su cui non effettuare i processi seguenti), per esempio: BANCA
31-60	30	No	Parola che sostituisce la parola eccezione, per esempio: BANCA

MCHR - Multiple characters

Col.	Dim.	Obbligatorio	Descrizione
1	1	Sì	Set caratteri, generalmente l'alfabeto, che, se doppi o tripli in sequenza all'interno di una parola, vengono ridotti ad uno solo

SUFX - Suffixes

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Suffisso da rimuovere

PRFX - Prefixes

Col.	Dim.	Obbligatorio	Descrizione
1-30	30	Sì	Prefisso da rimuovere

2.4 L'ambiente di sviluppo di Actr

Prima di entrare in merito sulle componenti specifiche dell'ambiente di sviluppo, è opportuno definire che cosa si intende per **CONTESTO** e per **PROGETTO** di codifica.

CONTESTO

Quando si utilizza Actr per assegnare un codice ad una descrizione testuale, sia l'insieme dei codici possibili, che il testo di input si rifanno ad una variabile ed alla relativa classificazione di riferimento, quali per esempio la Professione o l'Attività economica.
Un *"contesto"* Actr è l'insieme di informazioni di controllo e di file di dati che regolano la codifica di ciascuna variabile. Secondo l'esempio, avremmo quindi un contesto per la Professione ed uno per l'Attività economica.
Ogni contesto di codifica è costruito utilizzando specifiche e dati forniti dall'utente sotto forma di file testo. Questi file forniscono ad Actr le informazioni necessarie per standardizzare il testo ed assegnare il codice. Actr memorizza ed effettua ricerche sui contesti, utilizzando database interni per ogni contesto.
Fisicamente ogni ***contesto*** *consiste in un insieme di file -* ***Parsing Strategy file, Parsing Data file,*** **C*onfiguration file, Reference file*** *ed, infine****, il database Actr*** .
Per modificare un determinato contesto si può innanzitutto utilizzare l'editor per modificare uno o più dei file citati. Inoltre si può utilizzare Actr direttamente per aggiornare il contesto di interesse. Si può per esempio, come si vedrà in seguito, ricaricare (LOAD) il database e ricalcolare i pesi per le parole (PREP).

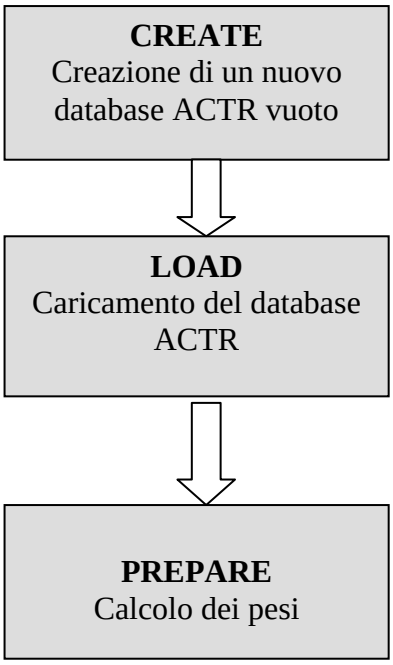
PROGETTO

Un progetto Actr è essenzialmente il set di risorse che costituisce l'intera applicazione di codifica.
Include uno o più contesti e strategie di codifica.
Fisicamente la definizione di un ***progetto*** *è memorizzata su un file di sistema,* che, in ambiente Windows NT, ha convenzionalmente il nome ***Project-name.GCP*****.**
Il *Project file* contiene le informazioni di definizione e controllo pertinenti al progetto, ma non i file del database.

2.4.1 I comandi Actr e la sequenza delle principali operazioni

Le fasi (ed i relativi comandi) da eseguire quando si costruisce un ambiente di codifica con Actr sono quelle riportate nella figura 3.

Figura 3 - Sequenza delle principali operazioni



La struttura tipo dei comandi Actr è la seguente:

«ACTR *command_name configuration_filename options*».

Le **risorse** necessarie per iniziare sono le seguenti:

- **Reference Data file(s)** – almeno due se si vuole un ambiente a contesti multipli (confrontare paragrafo 2.4.4.6 e paragrafo 2.4.4.8)
- **Parsing Data file(s)** (confrontare paragrafo 2.4.4.4)
- **Configuration file(s)** – uno per ogni contesto (confrontare paragrafo 2.4.4.1)
- **Spazio disco** – spazio sufficiente per caricare il/i database. Come indicazione di massima, si consideri un minimo di 1,5 Mb per ciascun contesto, ipotizzando di caricare, per esempio, 1.000 record di 100 byte ciascuno.

In generale, le **opzioni** associabili ai comandi sono le seguenti:

-L<c> lingua utilizzata nei messaggi
- -le → inglese (default)
- -lf → francese
- -lb → bilingua (entrambe)

-M<n> message level
- -M0 → nessun messaggio
- -M1 → messaggio sintetico
- -M2 → ulteriori messaggi di errore
- -M3 → ulteriori wording
- -M4 → ulteriori notifiche
- -M5 → messaggi più dettagliati

CREATE

Il comando CREATE crea un nuovo database Actr vuoto, oltre ad un file di report sull'esito dell'operazione (il cui nome è quello specificato nel *Configuration file*, linea REPORTPATH, confrontare paragrafo 2.4.4.1). Actr considera il nuovo database ed il suo ambiente come un contesto a sé stante. Infatti, se si prevede di costruire una procedura di codifica basata su più contesti (multi-context project, confrontare paragrafo 2.4.4.8), è necessario eseguire il comando CREATE per ciascuno dei contesti.
Actr crea il database utilizzando il nome specificato in corrispondenza del parametro del *Configuration file* CTXPATH.

La sintassi del comando è:

«ACTR CREATE *configuration_filename*»

Esempio:

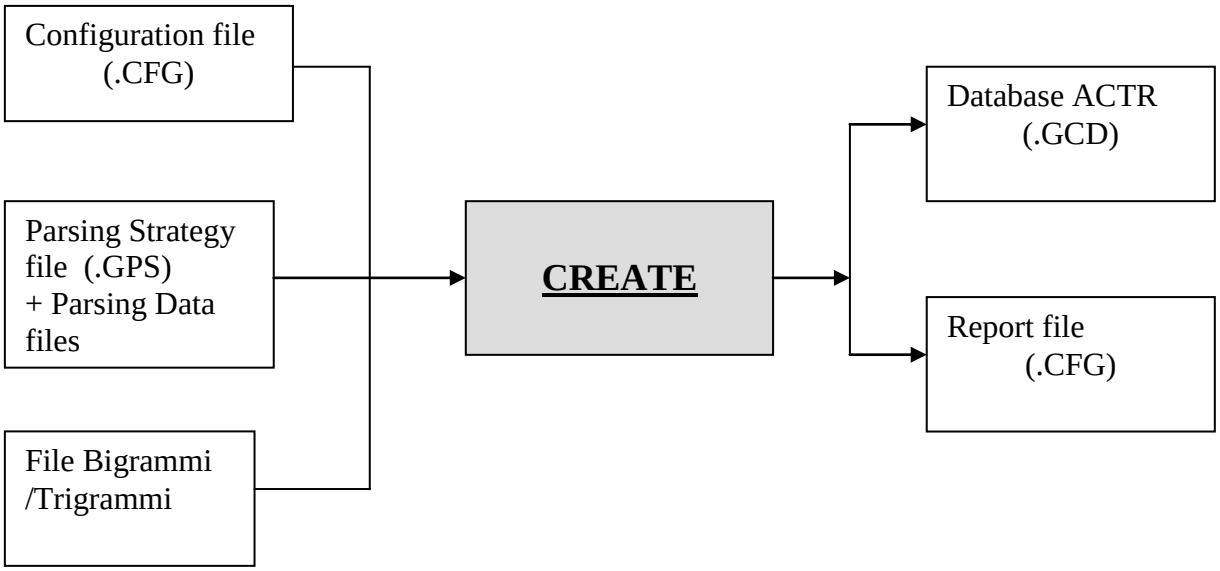
«ACTR CREATE C:\prof\prof01.CFG».

In ambiente Windows/NT il nome del database Actr ha per convenzione l'estensione **GCD**, mentre in ambiente UNIX l'estensione può variare.

È buona regola includere il nome del contesto nel nome della directory che lo contiene. È anche consigliato usare la stessa directory per il database e per il file di report della procedura di CREATE che lo ha generato.
I **prerequisiti** per effettuare il CREATE sono i seguenti, come riportato nello schema della figura 4:

- la directory specificata in CTXPATH, preferibilmente vuota,
- un valido *Configuration file*, con appropriati path e nomi di file,
- il *Parsing Strategy file*, con il nome riportato nel *Configuration file*, in corrispondenza di STRATPATH,
- i *file di parsing*, richiamati nello *Strategy file*,
- il file di Bigrammi/trigrammi, con il nome riportato nel *Configuration file*, linea GRAPHPATH.

Figura 4 - Prerequisiti per effettuare il CREATE



LOAD

Il comando LOAD consente di *"riempire"* il database Actr, precedentemente creato, con le descrizioni testuali ed i codici corrispondenti contenuti nel dizionario della classificazione (*Reference file*).

La sintassi del comando è:

«ACTR LOAD *configuration_filename.*»

Esempio:

«ACTR LOAD C:\prof\prof01.CFG».

I **prerequisiti** per effettuare il LOAD sono i seguenti:

- il *Reference file* contenente le stringhe di testo, i codici associati ed, eventualmente, il filtro,
- il *Configuration file*, che fornisce al LOAD le informazioni sulla struttura del *Reference file*, il suo contenuto e sugli altri file utilizzati da Actr,
- il *Parsing Strategy file*, con il nome riportato nel *Configuration file*, in corrispondenza di STRATPATH,

- i *file di parsing*, richiamati nel *Parsing Strategy file* ,
- il database Actr relativo al contesto in esame (vuoto).

Nel database Actr sono caricati soltanto i record del *Reference file* che il sistema accetta come validi. I record che non soddisfano i criteri di caricamento previsti di default dal sistema o dai filtri (confrontare paragrafo 2.4.4.7) sono registrati nell'apposito file dei record rifiutati (**Rejected Records** file).

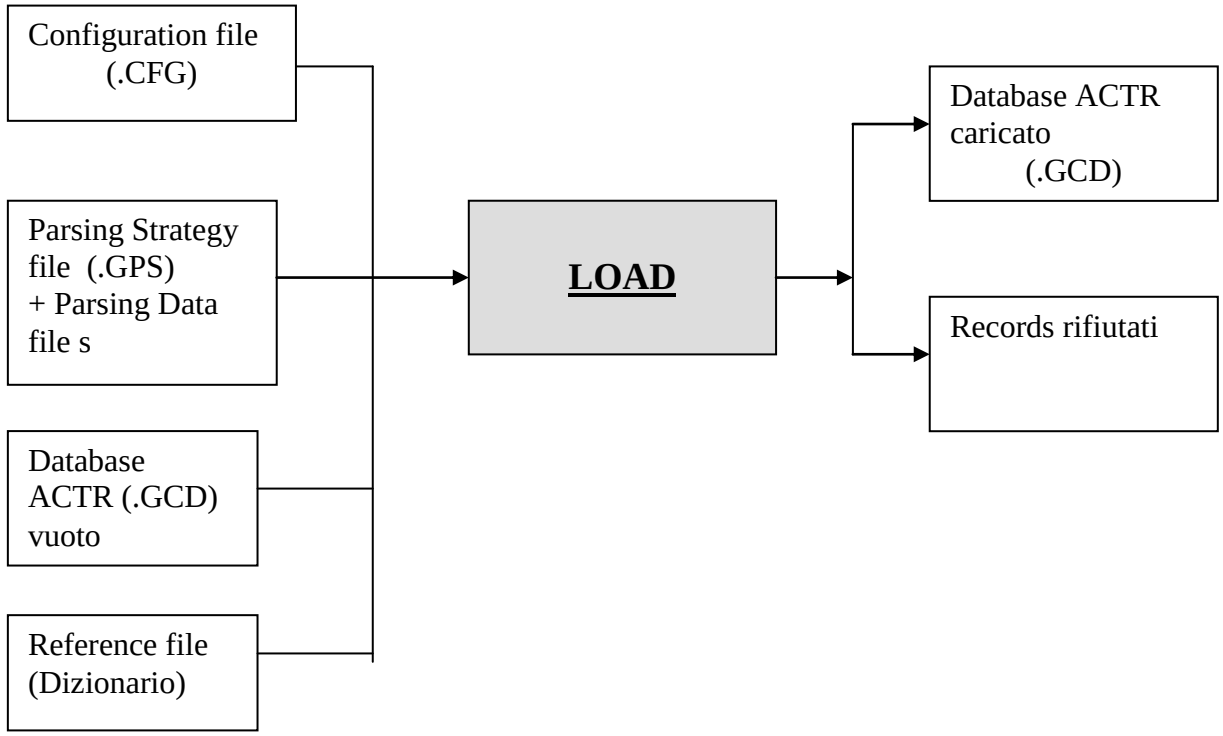
In particolare sono rifiutati i record considerati **duplicati**, in quanto:
- hanno entrambi la stessa descrizione dopo il *parsing*;
- hanno lo stesso codice ed eventualmente lo stesso filtro.

I **duplicati** sono consentiti soltanto se:
- hanno la stessa descrizione (dopo il *parsing*), ma codici diversi **e**
- contengono una **Deletion Clause.**

È opportuno infine specificare come funzionino le *"Deletion Clauses"* **nell'utility di LOAD.**
A volte, infatti, può sembrare che la *strategia di parsing* lavori in un modo troppo restrittivo.
Si possono utilizzare le *"Deletion Clauses"* per superare questo problema.
Ecco un esempio che illustra l'effetto delle *"Deletion Clauses"* e dei filtri.
Supponiamo di avere il *Reference* delle Professioni nel quale, in corrispondenza delle diverse categorie, è stato utilizzato il filtro. Nell'ambito di ciascun filtro, alcune descrizioni di professioni sono molto simili, tali che dopo il *parsing* divengono uguali, ma associate a codici diversi. L'utility di LOAD verifica il valore del filtro, del risultato del *parsing* e della *"Deletion Clauses"*, in fase di caricamento dei record. Se non c'è alcuna *"Deletion Clauses"*, se due descrizioni sono uguali dopo il *parsing*, carica soltanto la prima; questo potrebbe significare che il codice corrispondente alla seconda descrizione non verrebbe mai caricato nel database.
Per evitare questo, possono essere utilizzate le *"Deletion Clauses"* per significare che, nonostante la similarità tra i testi, le due descrizioni rappresentano in realtà diverse Professioni.
I file relativi alla fase di *parsing* (strategia e dati) sono utilizzati nel processo di LOAD per caricare nel database Actr le descrizioni del *Reference file* standardizzate.
Nella versione corrente del software la funzione di LOAD è unicamente di tipo batch. Per le versioni future si prevedere anche una versione interattiva per consentire all'utente di controllare, ed eventualmente correggere, i record rifiutati nel processo di caricamento.
Attualmente, invece, è necessario analizzare singolarmente i record contenuti nel file dei *"rifiutati"* (che è un file testo) per apportare successivamente le necessarie modifiche nel dizionario della classificazione.
Le descrizioni sono caricate come stringhe binarie, ossia come CPK (Compressed Phrase Keys), utilizzando (256-39)= 217 numeri binari liberi per byte per comprimere i bigrammi/trigrammi più frequenti.

I **prerequisiti** per effettuare il LOAD sono schematizzati nella figura 5:

Figura 5 - Prerequisiti per effettuare il LOAD



PREPARE

Il comando PREPARE è utilizzato per calcolare i pesi delle parole, sulla base della frequenza con cui queste sono presenti nel database. I risultati sono memorizzati in appositi file di sistema che completano il database Actr.
Riguardo agli algoritmi per il calcolo dei pesi, si veda il paragrafo 1.4.

La sintassi del comando è:

«ACTR PREP *configuration_filename*».

Esempio:

«ACTR PREP C:\prof\prof01.CFG».

2.4.2 Le TAG

Una TAG costituisce un identificativo (non definito dall'utente) del record, un numero sequenziale che può andare da 1 a *n*, dove *n* è il doppio del numero di record del database. Le TAG possono essere caricate esclusivamente su un database vuoto (nuovo).

L'utilizzo delle TAG è opzionale; non hanno effetto sulla fase di matching, né sull'attribuzione del codice, ma ne può essere effettuato il display da parte di vari componenti della "*Graphical User Interface*". Le TAG forniscono uno strumento per individuare un record nel database nel corso della vita di quest'ultimo. Rendono facile, per esempio, collegare un testo "*parsed*" a quello originario del *Reference file*, purché le TAG siano state previste nel *Reference file* e caricate (LOADED) nel database.
Idealmente sarebbe auspicabile che le TAG fossero costituite da una serie di numeri senza gap, ma in realtà i gap sono ammessi.
Qualora si voglia aggiungere record ad un database già esistente e dotato di TAG, è necessario scaricare il database, cancellarlo dalla directory, effettuare il merge tra il file scaricato e quello dei nuovi record, quindi ricaricare il nuovo *Reference file* in un nuovo database, che conterrà tutti i record.

TAG e LOAD utility

Se il *Configuration file* (confrontare paragrafo 2.4.4.1) specifica che una TAG è presente nel *Reference file*, l'utility di LOAD si aspetterà una TAG in ogni record di input. Dal momento che le TAG hanno valore soltanto qualora siano uniche, la LOAD effettua una serie di controlli.
In particolare vengono applicate le seguenti regole:

- una LOAD che utilizzi le TAG può essere effettuata solo su un contesto nuovo, viceversa una LOAD senza TAG può essere effettuata sia su un contesto nuovo che su uno vecchio;
- un record senza TAG sarà rigettato dalla LOAD, ma il caricamento sarà portato a termine;
- un record con una TAG che ne duplichi una esistente nel database sarà rigettato (se una TAG in input è minore del valore esistente della successiva TAG disponibile, si ricerca nel database un eventuale duplicato: se non c'è alcun duplicato, il record viene aggiunto nel database, oppure, in caso contrario, viene rigettato e compare un messaggio di errore);
- se la TAG in input è maggiore del valore esistente della successiva TAG disponibile, quest'ultimo viene impostato al valore della TAG di input + 1.

Quando l'operazione di LOAD è completata e nessun errore serio è stato diagnosticato, il valore esistente della successiva TAG disponibile viene impostato con il successivo valore disponibile.
I record appartenenti a ciascuna categoria – rigettati, caricati e così via – sono comunque scritti nel file di LOG e ne viene anche effettuato il display a schermo.

2.4.3 Altri comandi ACTR

Ai tre citati comandi, se ne aggiungono altri tre di utilità:

- **UNLOAD**
- **HELP**
- **SHELL**

Riferendoci in particolare all'utility **UNLOAD**, questa serve per estrarre dati dal context database di Actr. I dati vengono posti in un file sequenziale, nello stesso formato del *Reference file*.
UNLOAD utilizza il *Configuration file* per determinare la posizione e la lunghezza dei campi.

Se sono presenti le TAG, sono poste in output allineate a destra, premettendo alle cifre significative gli zeri, per esempio:

000001
000002

Ciò rende il file sequenziale di più facile lettura per l'utilizzatore, ma non ha niente a che fare con il modo in cui la TAG è memorizzata internamente; per esempio potrebbe essere stata memorizzata come
«1bbbbb» (dove b = blank).
A parte questa formattazione delle TAG, i campi in output saranno identici a come erano stati originariamente caricati dal *Reference file* nel database.
L'UNLOAD può essere molto utile per creare dei file di backup; ciò può essere preferibile rispetto a mantenere copie del *Reference file* originale che potrebbe contenere record rifiutati dalla fase di LOAD.
Il file prodotto dall'UNLOAD può essere ricaricato tramite l'utility LOAD.

La sintassi del comando è:

«ACTR UNLOAD *configuration_filename*».

Esempio:

«ACTR UNLOAD C:\prof\prof01.cfg».

L'utility UNLOAD leggerà prima il *Configuration file* per prendere le informazioni sul contesto da scaricare e sul *Reference file* da creare ed infine inizierà a scaricare i record dal database nel *Reference file*.
L'UNLOAD effettua questa operazione seguendo lo stesso ordine in base al quale il database era stato creato.
Relativamente all'**HELP** sulle funzioni Actr, è sufficiente digitare**: ACTR** oppure **ACTR HELP** dal prompt. Actr effettuerà il display delle funzioni e delle opzioni disponibili.
Il comando **SHELL**, infine, lancia una Actr *command shell*.

2.4.4 I file dell'ambiente di codifica

Per sviluppare una procedura di codifica con Actr sono necessari alcuni file, che contengono dati, informazioni di controllo sui dati, processi e ordine con cui eseguire questi stessi, eccetera
Si tratta principalmente di file in formato testo, che possono quindi essere facilmente controllati ed aggiornati dall'utente.

Tavola 3 - Elenco dei file utilizzati da Actr

Tipo	Nome	Formato	Definito dall'utente (sì/no)	Descrizione
Configuration file	Contextname.CFG	Testo	sì	Utilizzato per creare (CREATE) e caricare (LOAD) un database Actr
Project file	Projectname.GCP	Binario	sì	Definisce come Actr usi uno o più database per codificare
Coding strategy			sì	Non è un vero e proprio file, ma è memorizzato nell'ambito del Project file
Parsing strategy file	Contextname.GPS	Testo	sì	Uno per ogni lingua e applicazione
Parsing Data file	RSTR.GPD,DCLS .GPD, ecc. (massimo 12 files)	Testo	sì	Confrontare paragrafo 2.3
Diagraph/Trigraph data file	CPKCHRE.TXT (English) oppure CPKCHRF.TXT (French)	Testo	sì	Uno per ogni lingua. Finalità: ottimizzazione della compressione del database Actr
Reference file	Contextname.DAT	Testo	sì	Contiene le stringhe "testo-codice ed eventualmente il filtro da caricare nel database Actr
Internal Actr files		Binario	No	Vari files di sistema, principalmente il database Actr

N.B. - L'estensione **GPD** non è obbligatoria e può essere sostituita con **TXT**.
- Relativamente ai nomi di file riportati nella tavola 3, si è utilizzato lo standard di utilizzare più volte il Contextname e differenziare via via i file tramite l'estensione. Tuttavia è possibile utilizzare nomi diversi e sostituire le estensioni **GPS** e **DAT** con **TXT**.

2.4.4.1 Configuration file

Il *Configuration file* definisce un *"contesto"* di codifica per Actr, ovvero il dizionario di riferimento, le informazioni di controllo sui dati e quelle sulla procedura di codifica stessa. Il contesto è strettamente correlato alla variabile da codificare.
È utilizzato in tutte le fasi preliminari alla codifica (CREATE, LOAD, PREPARE, PARSING).

Questo file indica al sistema dove collocare e trovare il database Actr, i dati (il dizionario di riferimento), i file di *log* e di *repor*t, quali parametri usare in fase di caricamento del database e dove trovare i *file di parsing*.

Ogni contesto di codifica deve avere un suo *Configuration file*.

Il *Configuration file* è scritto in formato testo e può essere modificato utilizzando un qualunque editor ASCII compatibile.

Esempio di Configuration file.

```
// Parametri comuni a tutti i processi
//-------------------------------------------------------------------------------//
CTXSIZE = 10000      // Estimated number of records to be loaded
CTXPATH    = C:\Ateco\Context\Ateco_Ref.GCD                    // Context pathname.
CTXDESC    = "Test database per la variabile Ateco (con filtro)"      // Context description.
REPORTPATH = C:\Ateco\Context\Report.Txt                       // Name of report file.

//-------------------------------------------------------------------------------//
// Parametri utilizzati in fase di creazione (CREATE) del nuovo database ACTR  //
//-------------------------------------------------------------------------------//
GRAPHPATH = C:\ACTR v3\Cpk\CpkChrf.Txt                         // Digraph/Trigraph data
file.
STRATPATH = C:\ACTR v3\Parsing_ATECO\Strategy.TXT              // Parser strategy file.

//-------------------------------------------------------------------------------//
// Parametri utilizzati in fase di caricamento (LOAD) del Reference file.   //
// I dati numerici specificano le caratteristiche del Reference file.       //
//-------------------------------------------------------------------------------//
REFPATH   = C:\Ateco\Reference\Ateco_Ref.Txt                   // Reference file (input).
FAILPATH  = C:\Ateco\Context\Reject.Txt                        // Name of fail file
(output).
FIRSTREC  =   1                                                // First record to load.
LASTREC   =   0                                                // Last record to load.
CODEPOS   =   4                                                // Position of code field.
CODELEN   =   7                                                // Length of code field.
TEXTPOS   =   15                                               // Position of text field.
TEXTLEN   =  200                                               // Length of text field.
FILTERPOS =   1                                                // Position of filter.
FILTERLEN =   2                                                // Length of filter.
TAGPOS    =   0                                                // Position of tag field.
TAGLEN    =   0                                                // Length of tag field.
```

Le parole chiave utilizzate nel *Configuration file* non sono sensibili al maiuscolo/minuscolo (sono state scritte in maiuscolo soltanto per evidenziarle graficamente); sono comunque obbligatorie se le operazioni (di LOAD, CREATE) le richiedono.
Generalmente è più pratico avere un *Configuration file* completo, con tutte le parole chiave ed i relativi parametri.

Ogni dato corrispondente a ciascuna parola chiave, come si vede dall'esempio, deve essere scritto su una riga a sé stante; l'ordine in cui questi vengono riportati non è importante.
Si descrivono di seguito i vari componenti del *Configuration file*.

A) Parametri e componenti comuni a tutti i processi

Identificativo	Descrizione	Esempio
//(al margine sinistro)	Linea di commento Il testo alla destra è considerato commento	
//(in qualsiasi punto)	Commento per una parte di riga. Il testo alla sinistra di // è elaborato, quello alla destra è considerato commento	
CTXPATH	Nome completo del dababase, compreso il path	CTXPATH = C:\Ateco\Context\Ateco_Ref.GCD // Context pathname
CTXDESC	Descrizione libera del database; deve essere descritta tra virgolette	CTXDESC = «Classificazione delle Attività economiche» // Context description
REPORTPATH	Path e nome e del file di report	REPORTPATH = C:\Ateco\Context\Report.Txt // Name of Report file

B) Parametri utilizzati in fase di creazione (CREATE) del nuovo database Actr

Identificativo	Descrizione	Esempio
GRAPHPATH	Path e nome del file dati contenente Bigrammi/Trigrammi	GRAPHPATH = C:\ACTR v3\cpk\CpkChrf.Txt // Digraph/Trigraph data file
STRATPATH	Path e nome del file di strategia di parsine	STRATPATH = C:\ACTR v3\Parsing_ateco\Strategy.Txt // Parser strategy file
CTXSIZE	Numero approssimativo di record che si intende caricare nel database	CTXSIZE = 10000 //

C) Parametri utilizzati in fase di caricamento (LOAD) del Reference file

Identificativo	Descrizione	Esempio
REFPATH	Path e nome del file dizionario (Reference) da caricare nel database Actr	REFPATH = C:\Ateco\Reference\Ateco_Ref.Txt // Reference file (input)
FAILPATH	Path e nome del file dei rifiutati (duplicati) dalla fase di LOAD	FAILPATH = C:\Ateco\Context\Reject.Txt // Name of «fail file» (output)

C1) Caratteristiche del Reference file (se 0 → nessuna informazione)

Identificativo	Descrizione	Esempio
FIRSTREC	Primo record da caricare	FIRSTREC = 1 // First record to load
LASTREC	Ultimo record da caricare (se = 0 → carica tutto il file)	LASTREC = 0 // Last record to load
CODEPOS	Colonna di inizio del codice	CODEPOS = 4 // Position of code field
CODELEN	Lunghezza in byte del codice	CODELEN = 7 // Length of code field
TEXTPOS	Colonna di inizio del testo	TEXTPOS = 15 // Position of text field
TEXTLEN	Lunghezza in byte del testo	TEXTLEN = 200 // Length of text field
FILTERPOS	Colonna di inizio del filtro (si specifichi 0 se il filtro non è utilizzato)	FILTERPOS = 1 // Position of filter
FILTERLEN	Lunghezza in byte del filtro (si specifichi 0 se il filtro non è utilizzato)	FILTERLEN = 2 // Length of filter
TAGPOS	Colonna di inizio dell'etichetta (si specifichi 0 se il filtro non è utilizzato)	TAGPOS = 0 // Position of tag field
TAGLEN	Lunghezza in byte dell'etichetta (si specifichi 0 se il filtro non è utilizzato)	TAGLEN = 0 // Length of tag field

2.4.4.2 Project file

Il Project governa l'attività di codifica, nel senso che seleziona il/i contesto/i da usare e le strategie di codifica. Un progetto, infatti, può contenere più di un contesto, ma ogni singolo contesto può essere condiviso da più progetti.
La definizione del Project è fisicamente memorizzata nel *Project file* (*projectname*.GCP).
La definizione del progetto include:

- i *matching parameter* da utilizzare → parametri adottati per la codifica, quali i parametri soglia, il tipo di accoppiamento (diretto, indiretto), eccetera,
- i contesti di riferimento e l'ordine di accesso agli stessi,
- le *"Stopping Rules"* da utilizzare in caso di contesti multipli (confrontare paragrafo 2.4.4.8),
- le *shunt string* che identificano strategie alternative rispetto a quella standard. (confrontare paragrafo 2.4.4.7).

Actr supporta fino ad 8 strategie per progetto. È possibile creare fino a 7 strategie di codifica, oltre a quella *"standard"* (che è sempre disponibile e non può essere rinominata).
I nomi di ciascuna strategia nell'ambito di ciascun progetto devono essere univoci. È inoltre possibile sempre rinominarle, cancellarle, oppure abilitare ciascuna di esse perché sia autonomamente selezionata, utilizzando le SHUNT.

Quando si utilizza la GUI per codificare (confrontare paragrafo 2.5.2), una strategia esistente può essere selezionata manualmente. In BATCH, la strategia di default è quella STANDARD a meno che:

- non sia stato incluso nel file di input un campo SHUNT **e**
- non esista una strategia alternativa che
  - sia stata abilitata per l'auto-selezione **e che**
  - contenga una stringa SHUNT che realizza un match perfetto con il corrente record di input.

⇨ **Nonostante il *Project file* sia un file testo, è vivamente consigliato di crearlo e modificarlo utilizzando l'editor della GUI *(Graphical User Interface).***

Il *Project file* viene automaticamente generato in fase di creazione del progetto di codifica tramite l'interfaccia Windows di Actr.
È organizzato in sezioni, contraddistinte dalle parentesi quadre, per esempio: [Project]. Le parole chiave, come PROJNAME sono sulla sinistra e sono separate dai loro valori tramite spazi ed un segno di uguale (=). Linee o frasi che iniziano per // sono commenti e vengono quindi ignorati da Actr.

Si riporta di seguito un esempio di *Project file*

```
[Project]


[Match Files]
```

[Context]

[Strategy]

2.4.4.3 Parsing Strategy file

Del *parsing* si è già abbondantemente parlato nel paragrafo 2.3, in questa sezione ci si soffermerà sulla creazione, utilizzo e manutenzione della *strategia di parsing* da un punto di vista operativo.
Una *strategia di parsing* di default è fornita con Actr v3, ma l'utente dovrebbe considerarla soltanto come un *"modello"*, un punto di partenza, più che come un optimum di strategia.
Come può vedersi dall'esempio, il *Parsing Strategy file* contiene riferimenti (coppie keywords-filenames) ai *file dati di parsing* e ad altri elementi della strategia nel suo complesso.
Inoltre, la strategia specifica al *parser* quali operazioni eseguire ed in quale ordine, sia a livello di elaborazione di stringhe che di parole.
Come il *Configuration file*, il *Parsing Strategy file* è un semplice file testo ASCII, che può essere editato da normali editor di testi.

Generalizzando, si ricorda sulla base di quanto già detto nel paragrafo 2.3, che i processi associabili a ciascuna keyword sono i seguenti:

- PhraseProcess (1, 2, 3) → DCLS, DSTR, RSTR
- WordProcess (1,2,3,4,5,6,7,8) → HWRD, IWRD, RWRD, DWRD, EXCP, SUFX, PRFX, MCHR
- PostProcess (1,2) → RDUP, SORT
- Autotrim → Yes or No
- Trimleft → qualsiasi stringa di caratteri stampabili
- Trimright → qualsiasi stringa di caratteri stampabili
- Hiphen → qualsiasi singolo carattere stampabile

Come già detto, abitualmente l'estensione del *Parsing Strategy file* è .GPS, ma può essere anche TXT; il suo formato è il seguente:

Col.	Lunghezza	Richiesto?	Descrizione
1-20	20	Sì	Deve essere una keyword valida
21-147	127	Sì	Parametro associato alla keyword

2.4.4.4 Parsing Data file

Come già visto, ad ogni *Parsing Data file* corrisponde una sua keyword nel *Parsing Strategy file*.
È **obbligatorio** definire il WChrDATA file, includendovi tutti i caratteri validi.
Gli altri *Parsing Data file* sono tutti opzionali, ma, qualora siano specificati ne*l Parsing Strategy file*, devono essere costituiti i corrispondenti file e corredati di dati validi.
Qualora il singolo processo non sia utilizzato, non si deve riportarlo in corrispondenza dei phraseprocess, wordprocess o postprocess, ma bisogna comunque fornire la keyword (esempio Dclsdata) ed associarvi una stringa non completamente vuota, al posto del filename (si consiglia di specificare **nullfile**).
Si legga lo schema seguente, che rispetta la sequenza dell'esempio precedente.

Process Keyword	**Filename Keyword**	**Filename format**	**Descrizione**
DCLS	Dclsdsata	..path….\dcls.GPD	Deletion Clauses
DSTR	Dstrdata	..path….\dstr.GPD	Deletion Strings
RSTR	Rstrdata	..path….\rstr.GPD	Replacement Strings
DWRD	Dwrddata	..path….\dwrd.GPD	Double Words
HWRD	Hwrddata	..path….\hwrd.GPD	Hyphenated Words
IWRD	Iwrddata	..path….\diwrd.GPD	Illegal Words
RWRD	Rwrddata	..path….\rwrd.GPD	Replacement Words
EXCP	Excpdata	..path….\excp.GPD	Exception Words
SUFX	Sufxdata	..path….\sufx.GPD	Suffixes
PRFX	Prfxdata	..path….\prfx.GPD	Prefixes
MCHR	Mchrdata	..path….\mchr.GPD	Multiple Characters
(nessuno)	Wchrdata	..path….\wchr.GPD	Word Characterss

Sebbene non sia obbligatorio utilizzare i nomi citati per i *file di parsing*, è consigliato includere la keyword nel nome, per facilitarne la leggibilità e la manutenzione.
L'estensione **GPD** non è necessaria e può essere sostituita con **TXT**.

2.4.4.5 Digraph/Trigraph file

È questo un file di sistema di Actr che contiene 255 record, contenenti stringhe di due/tre caratteri.
È utilizzato internamente da Actr per comprimere le descrizioni testuali già sottoposte al *parsing*, così da generare le cosiddette CPKs (Compressed Phrase Keys)
Actr fornisce due file di bigrammi e trigrammi, uno per il francese (CPKCHRF.TXT) ed uno per l'inglese (CPKCHRE.TXT). La dipendenza dalla lingua deriva dal fatto che i record contenuti nei file sono in ordine decrescente rispetto alla frequenza in cui questi normalmente si presentano in ciascuna delle due lingue.
Nel *Configuration file* si specifica quale dei due file utilizzare in corrispondenza del parametro GRAPHPATH (confrontare paragrafo 2.4.4.1).

⟹ **Si raccomanda vivamente di non modificare questi file.**

2.4.4.6 Reference file

Il *Reference file* è un file sequenziale che contiene stringhe di testo, codici assegnati alle stringhe e, opzionalmente, filtri.
La costruzione del *Reference file* esula dalle competenze del sistema Actr e deve avvenire secondo quanto esposto nel paragrafo 1.3.1.
Il formato del *Reference file* varia a seconda della lunghezza dei testi da considerare e dei codici.
La specifica del formato è effettuata nel *Configuration file*. Relativamente al campo opzionale, chiamato TAG, si rimanda al paragrafo 2.4.2.
La lunghezza massima dei record del *Reference file* è di **2.500 byte,** tuttavia Actr è in grado di elaborare testi con lunghezza massima di **200 byte**.

Field Name	Descrizione	Lunghezza
REFTEXT	Descrizione testuale a formato libero	Max 200 byte
REFCODE	Codice associato alla descrizione	Max 10 byte
REFFILTER	Filtro (opzionale)	Max 10 byte

Non è richiesto l'utilizzo di blank e separatori tra descrizione testuale, codice e filtro, ma bisogna tenere presente che questi facilitano la leggibilità.

Si riporta, di seguito, come esempio di *Reference file,* uno stralcio tratto dall'applicazione sull'Attività economica.

FILTRO	**CODICE**	**TESTO**
		AGRICOLTURA
		AGRICOLTURA, CACCIA E RELATIVI SERVIZI
		CACCIA

		CACCIA E RELATIVI SERVIZI
		COLTIVAZIONE AGRICOLA
		COLTIVAZIONI AGRICOLE; ORTICOLTURA, FLORICOLTURA
		PRODUZIONE DI CEREALI (COMPRESO IL RISO)
		PRODUZIONE DI FRUMENTO DURO
		PRODUZIONE DI FRUTTI OLEOSI
		COLTIVAZIONE DI BARBABIETOLA DA ZUCCHERO
		COLTIVAZIONE DI TABACCO
		COLTIVAZIONE DI ALTRI SEMINATIVI
		ESTRAZIONE DI CARBON FOSSILE, LIGNITE E TORBA
		ESTRAZIONE E AGGLOMERAZIONE DI CARBON FOSSILE
		ESTRAZIONE E AGGLOMERAZIONE DI LIGNITE
		ESTRAZIONE E AGGLOMERAZIONE DI TORBA

2.4.4.7 Utilizzo del filtro e dei campi SHUNT

Il campo **filtro** non è obbligatorio, ma, se utilizzato, consente all'utente di definire sottoinsiemi di dizionario in cui restringere la ricerca da parte del sistema ai fini dell'assegnazione del codice.
Per esempio, rifacendosi al caso dell'attività economica, il dizionario potrebbe essere suddiviso in tante categorie, cui corrisponderebbero altrettanti valori del filtro, corrispondenti ai rami principali della classificazione (agricoltura, estrazione, industria, commercio, eccetera).
In fase di codifica, quindi, se anche ai record da codificare sono stati assegnati gli stessi filtri, quando il sistema dovrà codificare attività relative per esempio all'agricoltura, effettuerà la ricerca nel dizionario limitatamente alla sezione cui corrisponde il filtro dell'agricoltura.
Se due filtri si differenziano soltanto per il numero degli spazi, Actr li considera uguali.
Per esempio:
« AUTO » = « AUTO ».

Se il campo **filtro è vuoto** (in corrispondenza di uno o più record del dizionario, oppure della risposta da codificare) Actr si comporta come per la codifica senza filtro, quindi effettua la ricerca su tutto il *Reference*.
Lo **SHUNT** fornisce un metodo per utilizzare una particolare strategia di codifica piuttosto che un'altra. Lo SHUNT è una stringa specificata e memorizzata in una *strategia di codifica* e che può inoltre essere memorizzata nel file di input. Se tale campo è previsto nel file di input, ogni record in cui tale campo è valorizzato sarà automaticamente codificato utilizzando la strategia di codifica con il valore dello SHUNT che realizza un match perfetto con quest'ultimo.
Gli SHUNT nelle strategie definite dall'utente devono essere univoci. Ciò è valido anche nel caso di stringhe SHUNT costituite da blank: per ciascun progetto ne è ammessa soltanto una.

2.4.4.8 Codificare con contesti multipli

Actr supporta l'utilizzo di un massimo di 8 *contesti* da consultare nell'ordine specificato dall'utente nella *strategia di codifica*. Inoltre, come già detto, può essere utilizzata più di una *strategia di codifica*. Ogni *strategia* contiene una *"Stopping Rule"*.
Le *"Stopping Rule"* possono essere specificate tramite la GUI (Project settings).
Lavorando a contesti multipli, quindi, oltre all'ordine in base al quale consultare i contesti, bisogna definire una regola in base alla quale Actr interrompe la ricerca per la codifica di un certo testo di input quando il risultato del match nel contesto corrente verifica alcune condizioni: è questa la *"Stopping Rule"*. Nella sua forma più semplice la *"Stopping Rule"* può corrispondere ad una delle condizioni di *"Unico"*, di "*Multiplo"*, di *"Possibile"*, di *"Nessuno"*.

- **Unico** → ricerca nei contesti successivi a quello in esame soltanto quando in questo stesso non viene individuato nessun testo che realizza un match *"Unico"*.
- **Multiplo** → ricerca nei contesti successivi a quello in esame soltanto quando in questo stesso non viene individuato almeno un testo che realizza un match *"Multiplo"*.
- **Possibile** → ricerca nei contesti successivi a quello in esame soltanto quando in questo stesso non viene individuato almeno un testo che realizza un match *"Possibile"*.
- **Nessuno (no stop)** → effettua la ricerca nei vari contesti, senza interrompersi, considerandoli quindi come fossero un unico contesto.

In ogni caso, il record selezionato è quello che ha un punteggio almeno a livello vincente; bisogna comunque tenere a mente che l'insieme dei parametri di Actr interagisce con la *"Stopping Rule"* definita. È quindi consigliabile, per testare la condizione migliore, fare più prove utilizzando le potenzialità della GUI (confrontare paragrafo 2.5.2).
Utilizzando le API è possibile costruire applicazioni multi-context in cui vengono definite *"Stopping Rule"* differenti da quelle citate.

Attenzione!
Abilitare una *"Stopping Rule"* potrebbe implicare che qualche *contesto* rimanga non consultato: per esempio, un match *"Unico"* in un *contesto* successivo a quello corrente potrebbe non essere preso in considerazione se la condizione della *"Stopping Rule"* fosse *"Multiplo"*.
Per questo motivo, non sono consigliate *"Stopping Rule"* ad un livello inferiore a *"Multiplo"* e si preferisce invece adottare la condizione *"Unico"*, oppure *"Nessuno"*.
Nella pratica quindi si consiglia di effettuare più test per ottenere risultati di codifica ottimali.

2.4.4.9 Altri file di sistema

Reject File (Fail File)

Questo file ha lo stesso formato del *Reference file*, così come è stato definito nel *Configuration file.*
È un file testo che può essere visualizzato o editato con qualsiasi editor di testi standard. Il nome attribuito a questo file è quello definito in corrispondenza del FAILPATH del *Configuration file* (confrontare paragrafo 2.4.4.1).
In Actr v3, questo file fornisce all'utente lo strumento per identificare i record errati, così da correggerli. A tale scopo, può essere utilizzato come input del LOAD in modo da poter aggiungere direttamente i record corretti nel database. Le prossime versioni di Actr prevederanno una funzione interattiva che realizza questa operazione.

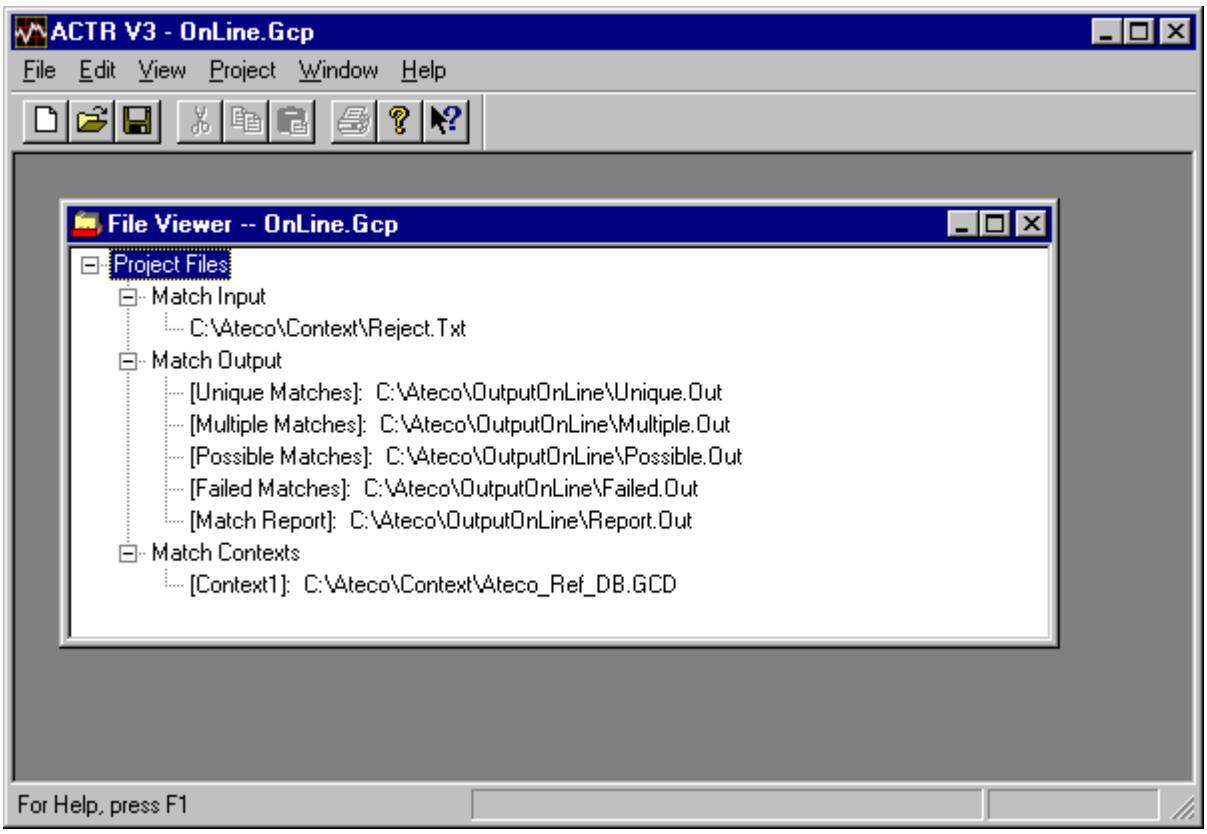
Database Actr

Internamente, il database Actr è un database relazionale. Per ciò che riguarda l'utente, l'utility di LOAD tratta il database come un'unica *"scatola nera"*, cui non si può accedere direttamente come ad un qualsiasi database.

2.4.5 Graphical User Interface (GUI)

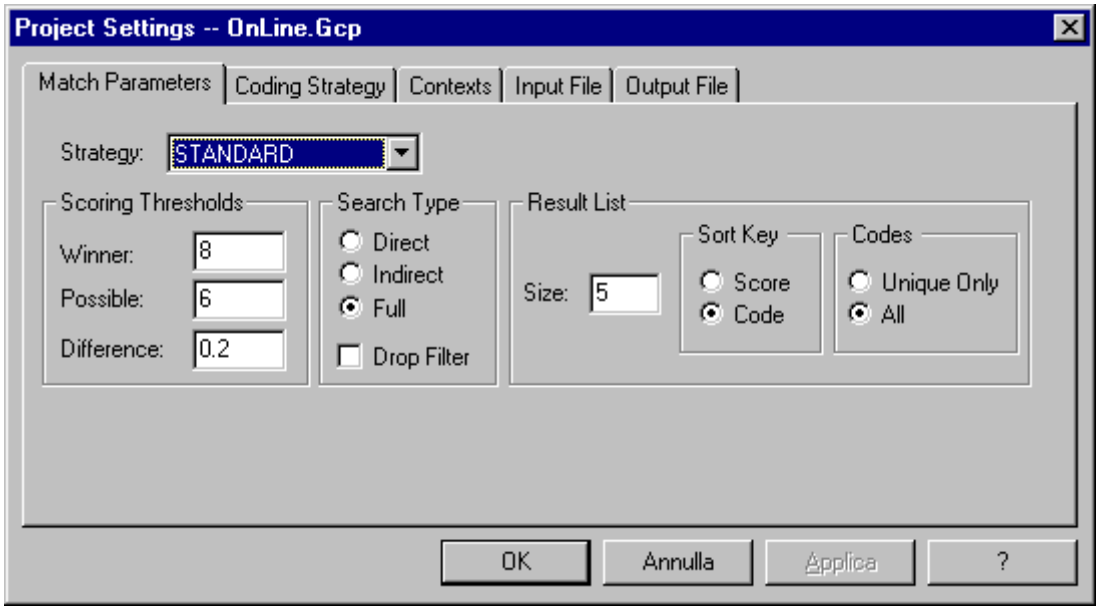
La Graphical User Interface di Actr costituisce un complemento, ma non può sostituire completamente il Command Line Interface (CLI). Infatti è spesso conveniente utilizzarli entrambi per realizzare tutto il processo di codifica (creazione dell'ambiente di codifica, sua manipolazione e codifica dei dati).
Si riporta di seguito la schermata principale della GUI, che permette la manipolazione del Progetto di codifica, consentendo di creare un progetto nuovo, modificarne uno esistente e gestire il matching online tra i dati da codificare ed il database Actr.
Altre funzioni, quali il CREATE, il LOAD ed il PREP saranno aggiunte tra le funzionalità della GUI nei prossimi release del sistema.
La figura 6 riporta la prima schermata che appare quando si lancia interattivamente Actr e si seleziona un progetto.

Figura 6 - Selezione del progetto



Selezionando il pulsante "PROJECT", e quindi "settings", si accede alle funzioni del Project editor e viene proposta la schermata di cui alla figura 7. Tramite questo editor viene messa a disposizione una serie di bottoni da selezionare che consentono di combinare contesti, strategie di codifica e di specificare i parametri di match ed i file da utilizzare. Ovviamente, il contesto deve già esistere per poter essere incorporato in un progetto

Figura 7 - Schermata principale del Project editor



In particolare, come può vedersi dalle figure riportate:

- selezionando il pulsante "`Match parameters`", è possibile impostare i parametri della strategia (figura 7)
- selezionando il pulsante "`Coding strategy`", è possibile definire le regole di accesso ai contesti (figura 8)
- selezionando il pulsante "`Context`", è possibile selezionare i contesti (figura 9)
- selezionando il pulsante "`Input file`", è possibile selezionare il file da codificare (figura 10)
- selezionando il pulsante "`Output file`", è possibile specificare le destinazioni dei file di output (figura 11)

Per il dettaglio sulla gestione di queste funzioni, si rimanda la paragrafo 2.5.2.
La GUI rende quindi semplice lo sperimentare differenti combinazioni di parametri e strategie di codifica, grazie al *"Match Manager"* interattivo che consente di verificarne subito gli effetti.

Figura 8 - Definizione della strategia di codifica

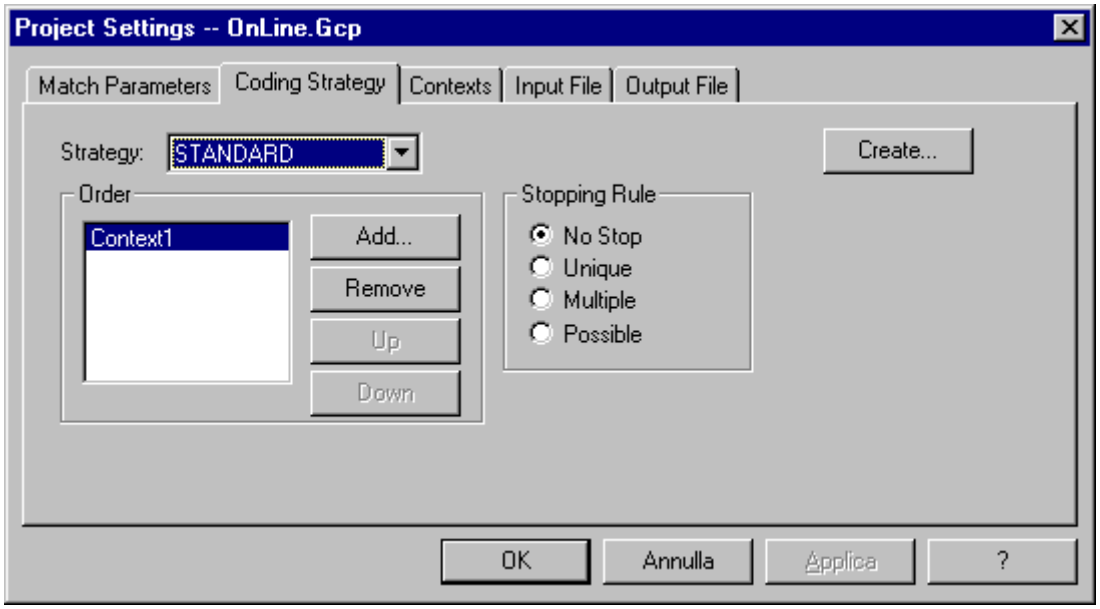


Figura 9 - Specifica dei contesti per il progetto

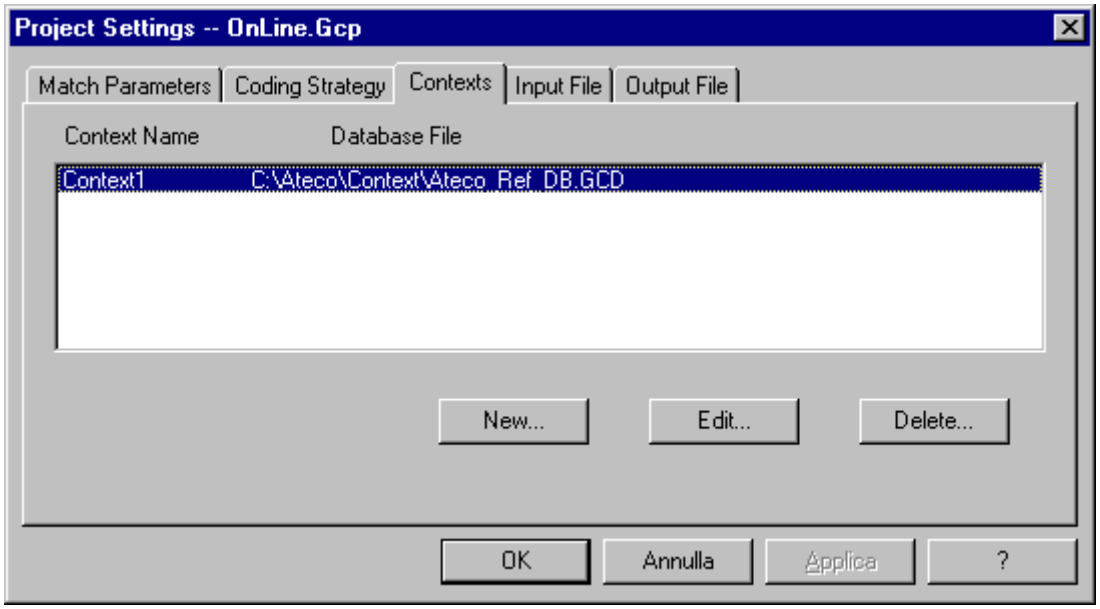


Figura 10 - Specifica del file di input da codificare

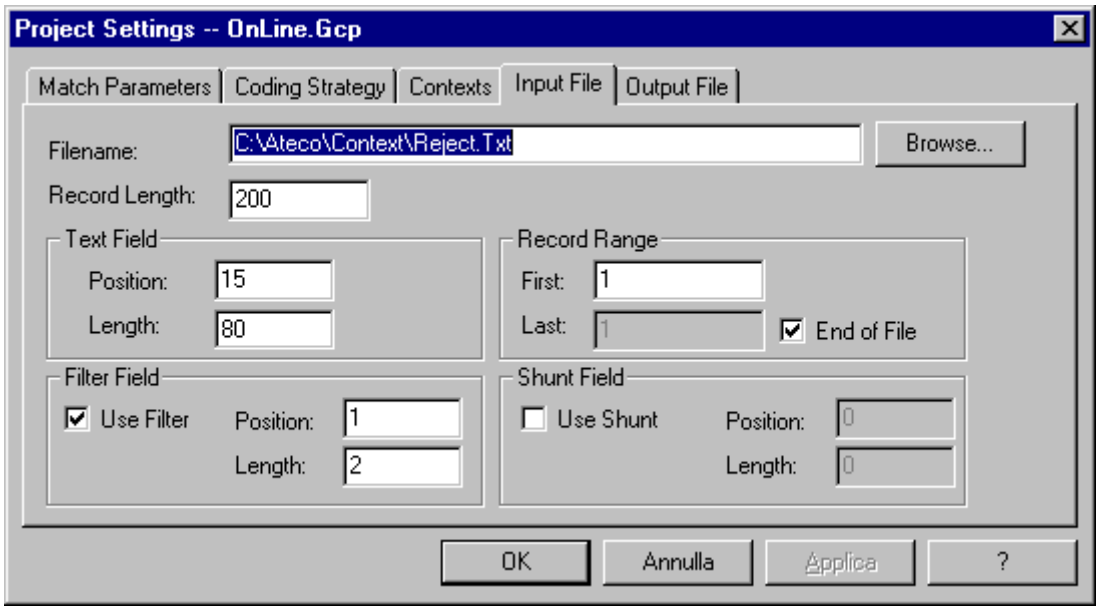
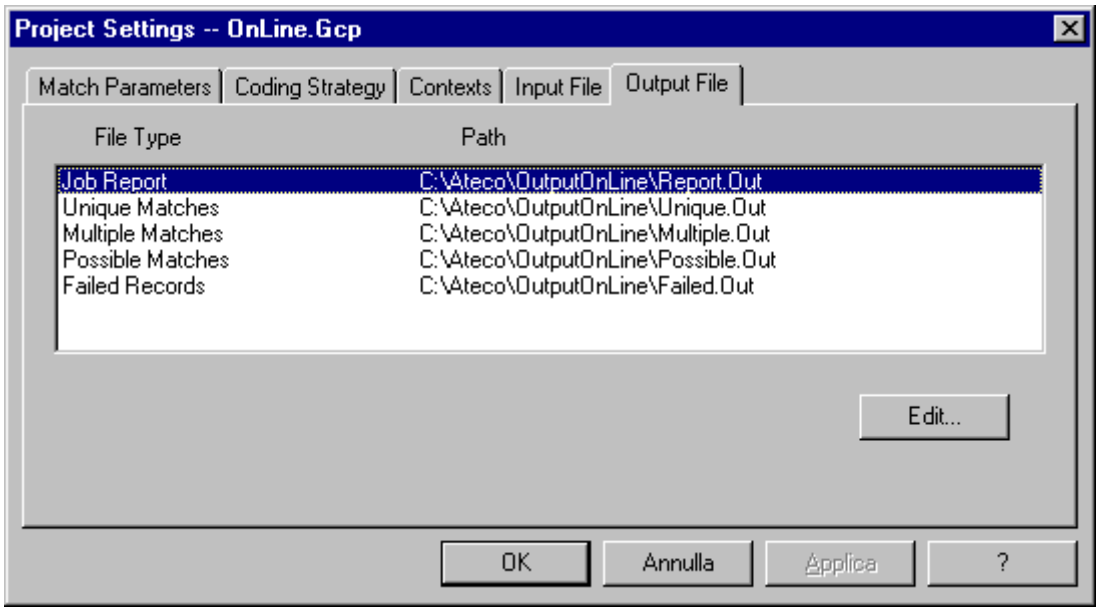


Figura 11 - Specifica delle destinazioni dei file di output



2.5 Codificare con Actr

La sequenza di operazioni da eseguire per codificare è la seguente:

1) modificare il *Configuration file* in modo che i nomi di path e di file siano coerenti con l'applicazione che si sta realizzando (processo di editing manuale),
2) creare il database vuoto (dal prompt: ACTR CREATE…),
3) caricare il database (dal prompt: ACTR LOAD…),
4) preparare il database per la codifica (dal prompt: ACTR PREP…),
5) definire un progetto con uno o più contesti (tramite la GUI), ricordando che un contesto già deve esistere, prima della creazione del progetto che lo utilizzerà,
6) definire la strategia di codifica (tramite la GUI) ricordando che, se si vuole, è possibile utilizzare quella STANDARD.

È ora possibile codificare:

7) effettuare la codifica (tramite il BCODE; è possibile effettuare il matching tra i testi da codificare ed il database ACTR tramite la GUI),
8) verificare come il Parser di Actr modifica le stringhe di testo da codificare (dal prompt: BPARSE.EXE; è possibile verificarlo online per ciascuna stringa tramite la GUI).

Qualora l'applicazione di codifica sia stata inserita in un'applicazione scritta in "C", che richiama Actr tramite le API, la relativa esecuzione avverrebbe inserendo il comando BCODE nel programma "C".
Come si vedrà nella pratica, utilizzare Actr è un processo iterativo per affinamenti successivi – si carica il *Reference file* con una certa strategia di codifica, si valutano i risultati dell'applicazione di codifica, si modifica la strategia in funzione di questi, si carica di nuovo e così via.
Ciò che è importante ricordare è che, le operazioni di caricamento e codifica richiedono entrambe gli stessi *file di parsing*, per assicurare la consistenza dei risultati. Ogni qual volta si modificano i *file di parsing*, sarebbe corretto cancellare il database e nuovamente crearlo, caricarlo ed effettuarne il PREP.

2.5.1 Codificare con il BCODE

Come già detto, il BCODE è un processo batch che si esegue dal prompt; lavora riferendosi al Project Actr, utilizzandone gli input, gli output e la strategia di codifica.
Nella pratica, è sempre consigliabile combinare gli approcci interattivo e batch in fase di sviluppo dell'ambiente di codifica, creando il progetto e valutandone i risultati interattivamente, tramite la GUI, quindi procedendo alla codifica vera e propria del file di input tramite il BCODE.
BCODE richiede che sia già stato definito il *Project file*, in cui è stato specificato il/i contesto/i da utilizzare, la strategia di codifica, eccetera.
Nel caso in cui si debbano codificare diversi campi, per ciascuno dei quali potrebbe essere richiesto di accedere a più contesti (per esempio i campi si riferiscono a variabili diverse), è preferibile creare applicazioni personalizzate utilizzando le API.

La sintassi del comando è:

«C\ACTR V3\BIN\BCODE <projectfile> [option1, option2….]»

dove :
- <projectfile> è il nome del progetto ed il relativo path,
- le opzioni relative ai livelli di messaggistica sono:
  - -M0 → nessun messaggio
  - -M1 → messaggi minimi
  - -M2 → messaggi di base
  - -M3 → messaggi di sintesi
  - -M4 → ulteriori messaggi di errore
  - -M5 → ulteriori wording
  - -M6 → ulteriori notifiche
  - -M7 → messaggi più dettagliati
  - -M8 → altri messaggi.

Esempio:

«C\ACTR V3\BIN\BCODE\COD_ATECO\ATECO.GCP»

Si riporta di seguito ciò che compare a schermo, a seguito dell'esecuzione del BCODE.

```
C:\ACTR V3\BIN\BCODE C:\ACTR V3\COD_ATECO\ATECO.GCP
```

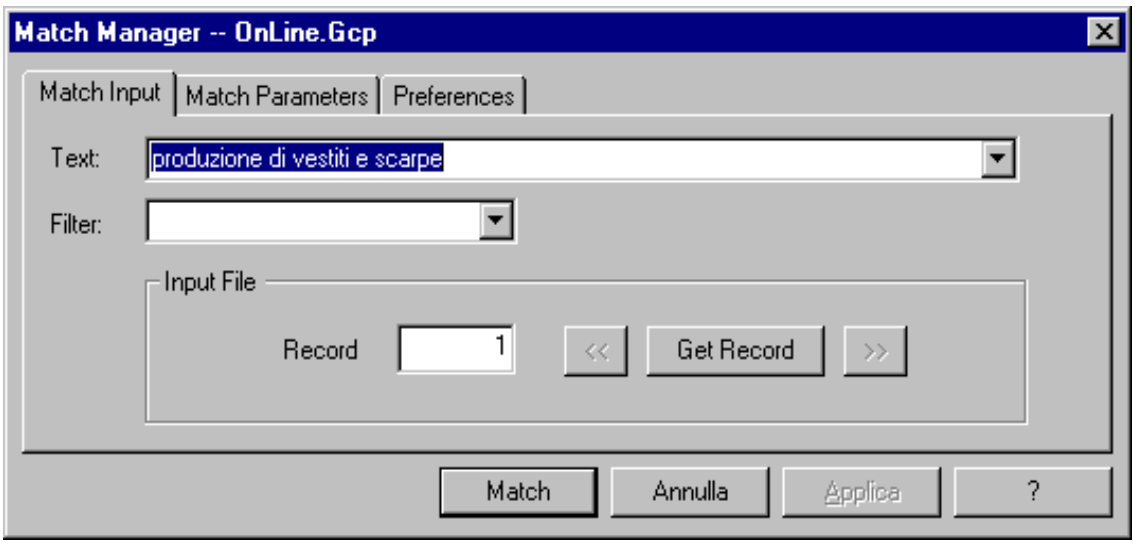
"Overflowed" significa che, nel tentativo di abbinamento, il criterio di matching è stato soddisfatto da troppi record, così Actr non è stato in grado di fornire un codice.
Relativamente ai due *"Coding summary"*, il primo è il più indicativo, in quanto riporta la distribuzione dei record da codificare nei file di output. Il secondo, invece, calcola le percentuali sui record generati a seguito della codifica.

2.5.2 Codificare dalla GUI

Codificare interattivamente dalla GUI è estremamente semplice. Dal menu *"Match manager"*, ci sono tre pulsanti che consentono rispettivamente di:

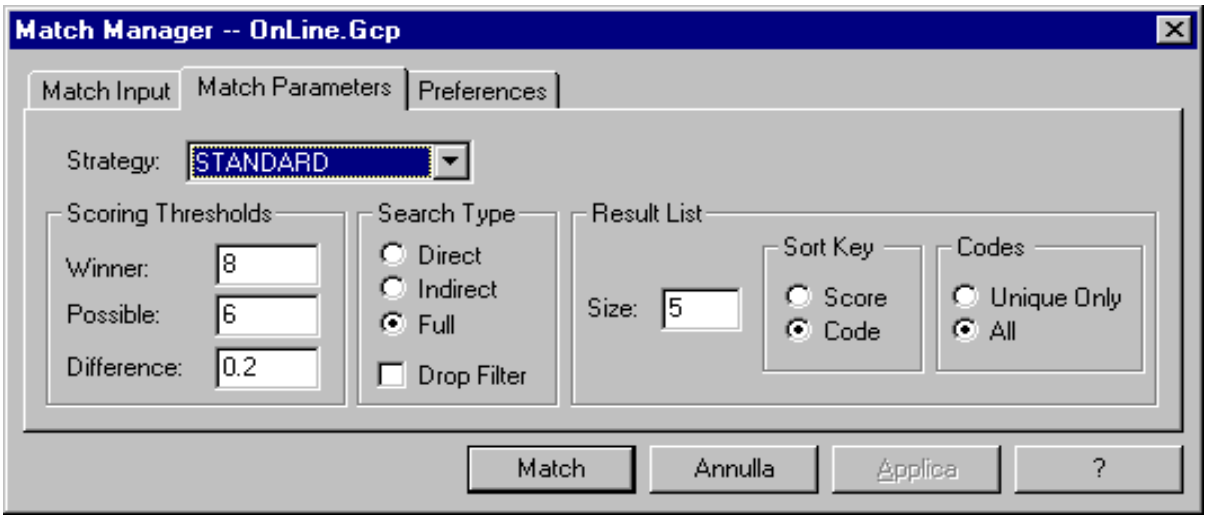
- definire l'input da codificare,
- definire i parametri di sistema,
- modificare la modalità di display a schermo delle finestre.

Figura 12 - Definizione dell'input da codificare



In questa finestra è possibile fornire il testo da codificare in tre modi: digitando il testo nell'apposito box, richiamandolo dal file di input nella sequenza in cui si presentano i record, richiamandolo dal file di input, selezionando il numero di record corrispondente.

Figura 13 - Definizione dei parametri di sistema



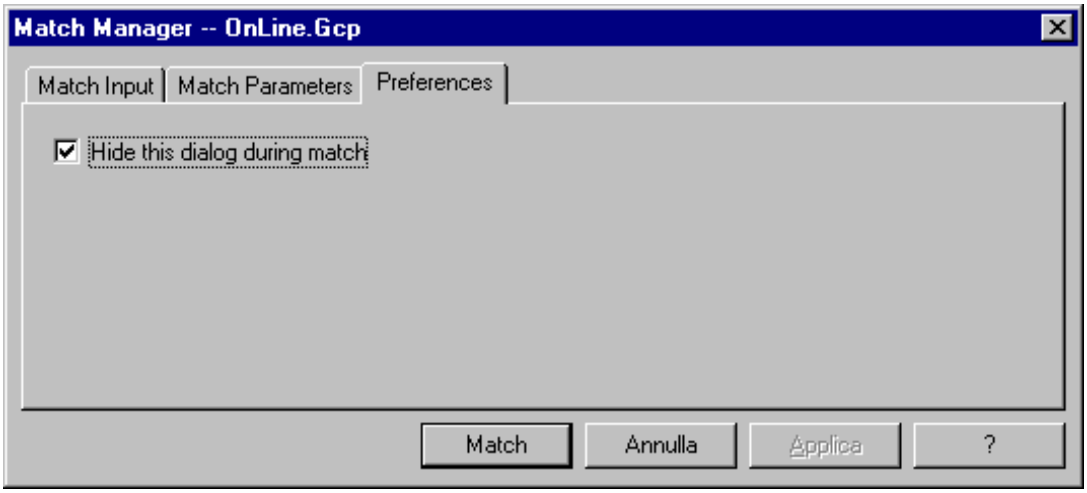
Tramite questa finestra è possibile definire una serie di parametri.
Innanzi tutto la **strategia**. Nell'ambito di quella **STANDARD,** nel box *"Scoring thresholds"*, è possibile modificare le soglie che, come riportato nel paragrafo 1.4.1, corrispondono a Smax, Smin e ΔS (i valori di default dei parametri soglia sono quelli riportati nella figura 13). È inoltre possibile selezionare un'altra strategia, la **DEEP**, che è molto utile in fase di test, oppure qualora si voglia navigare nel dizionario in modo molto elastico, in quanto consente di estrarre dal dizionario tutti i testi che abbiano almeno una parola in comune con il testo da codificare, indipendentemente dalle soglie.
Il box "*Search type"*, invece, serve per richiedere ad Actr di codificare secondo uno dei tre seguenti metodi di matching:

- **DIRETTO** → vengono estratti dal database soltanto i testi identici, a seguito del *parsing*, a quelli da codificare;
- **INDIRETTO** → vengono utilizzati i pesi delle parole ed i punteggi finali, per misurare la similarità tra i testi del database e quelli da codificare;
- **FULL** → Actr tenta prima di individuare un match diretto, quindi, in caso di insuccesso, tenta quello indiretto (FULL è il parametro di default previsto dal sistema).

Inoltre, è possibile specificare come il sistema si debba comportare qualora si codifichi con il filtro. In particolare, se non si seleziona il box **"DROP FILTER"**, Actr, se a seguito del tentativo di match con il filtro fallisce, interrompe la ricerca. In caso contrario prosegue la ricerca su tutto il *Reference* a prescindere dal filtro (l'opzione DROP costituisce il default previsto da Actr).

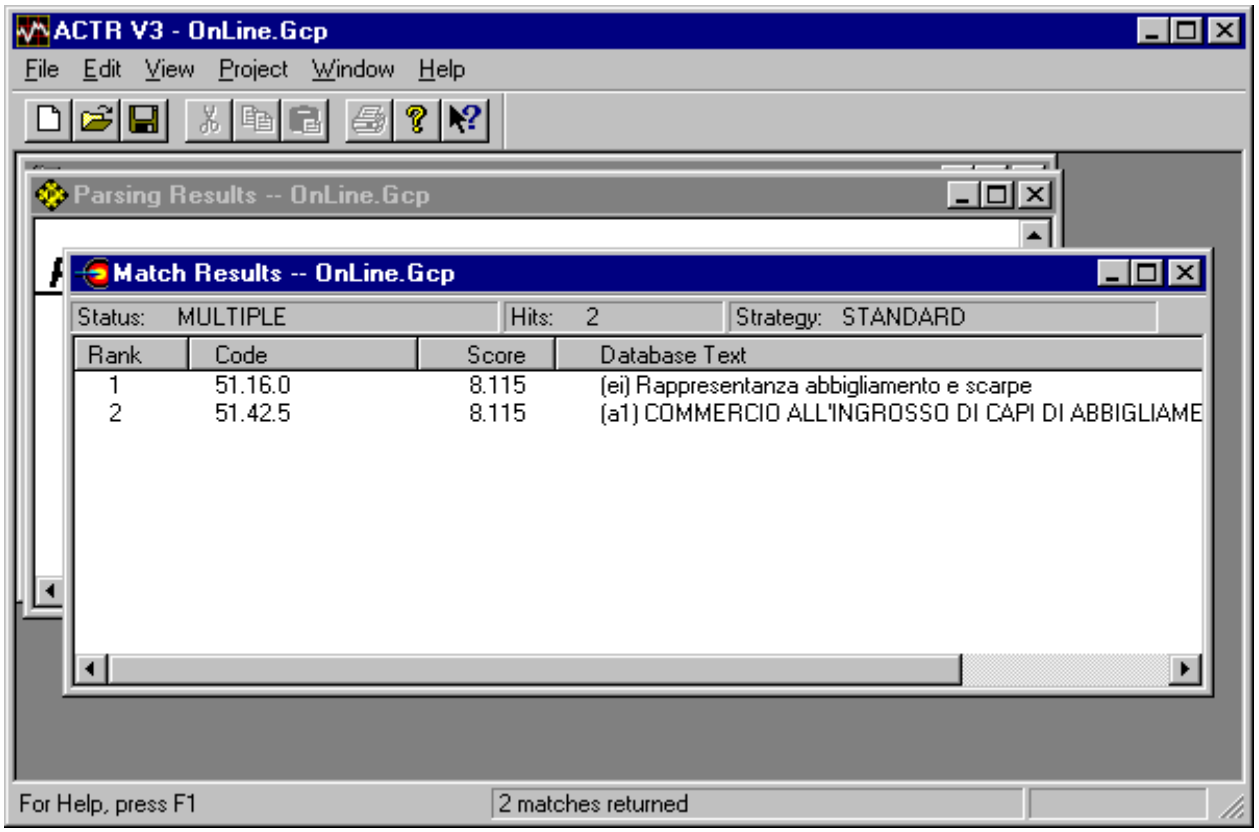
Nel box "*Result list"* si specifica invece:
- **«SIZE»** → il numero massimo di match da produrre in output (nel caso di *"Multipli"* e *"Possibili"*); il valore di default è 5,
- **«SORT KEY»** → il criterio di ordinamento dei risultati, per punteggio o per codice, (sempre nel caso di *"Multipli"* e *"Possibili"*); il criterio di default è il punteggio,
- **«CODES»** → cosa produrre in output, solo gli Unici oppure tutti e quattro i file; il default è Unique.

Figura 14 - Modifica della modalità di display a schermo delle finestre



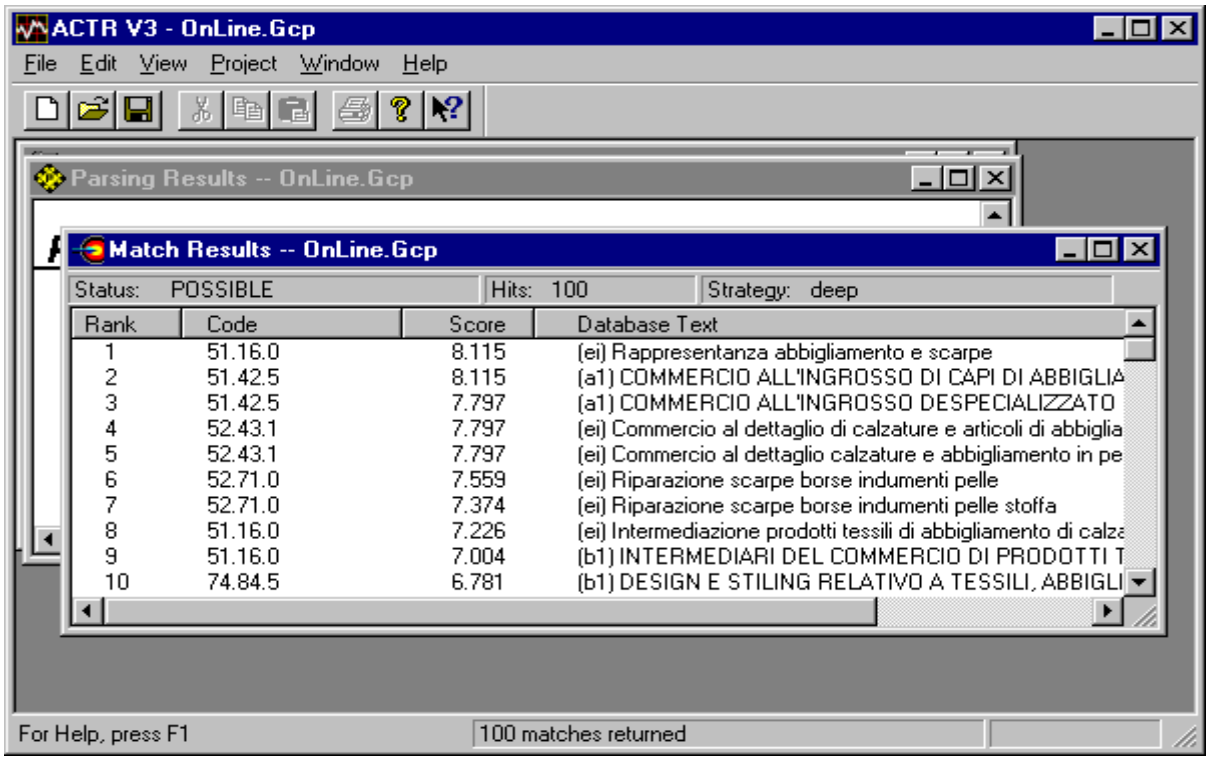
La terza finestra consente invece di ricoprire, con la finestra in cui sono riportati i risultati della codifica, quella iniziale (figura 12).
Il risultato del matching interattivo viene riportato tramite la schermata della figura 15; come si può vedere, a titolo puramente esemplificativo, il testo «PRODUZIONE DI VESTITI E SCARPE» presenta un elevato livello di similarità con due testi del dizionario: «RAPPRESENTANZA DI ABBIGLIAMENTO E SCARPE» e «COMMERCIO ALL'INGROSSO DI CAPI DI ABBIGLIAMENTO». Entrambi questi testi, però, sono associati ad uguale punteggio, quindi, non essendo superata la soglia ΔS, il risultato della codifica rientra nei *"Multipli"*.

Figura 15 - Risultato del matching interattivo (strategia STANDARD)



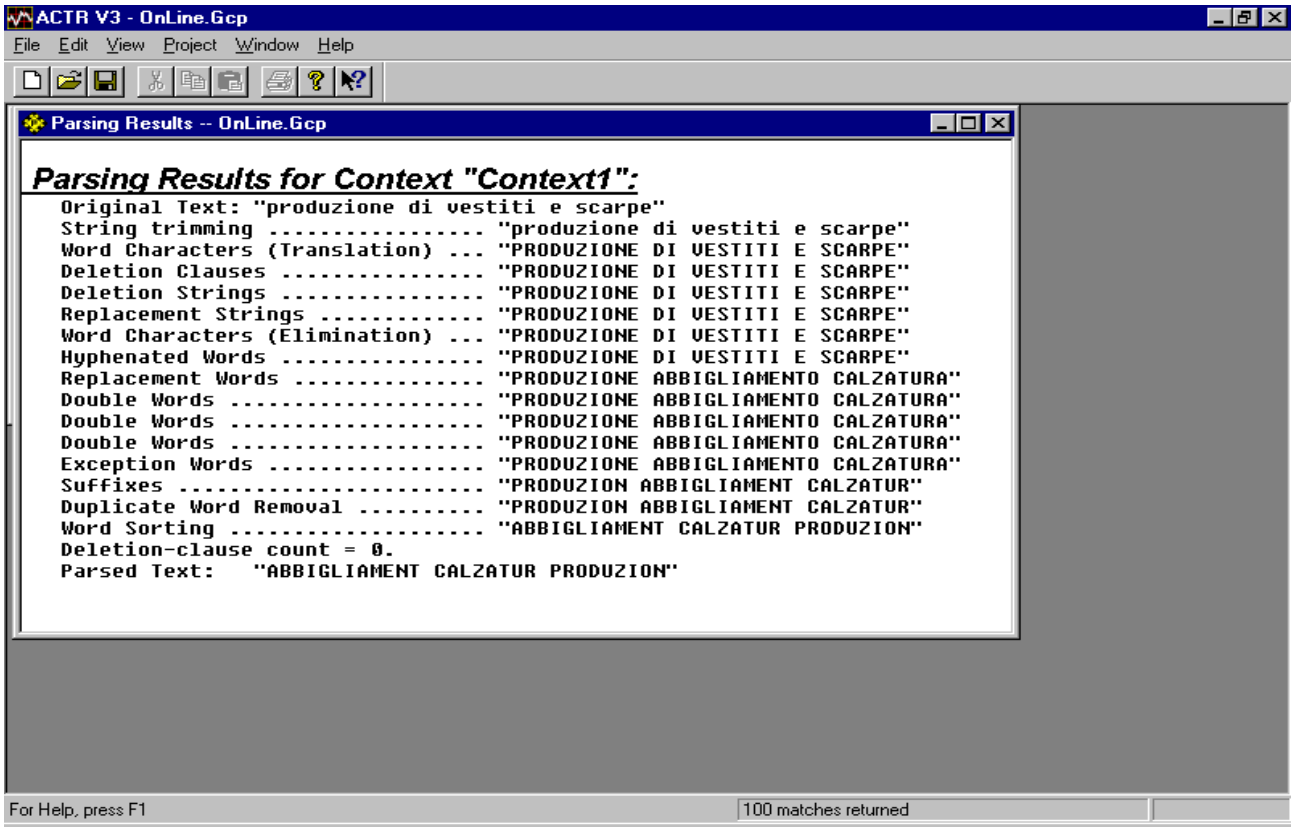
Qualora fosse stata impostata la strategia DEEP, anziché quella STANDARD, il risultato sarebbe stato quello riportato in figura 16.

Figura 16 - Risultato del matching interattivo (strategia DEEP)



È possibile verificare anche come il *parsing* abbia elaborato il testo di input, "cliccando" sulla finestra "Parsing results" (sottostante rispetto a quella "Match results").
Rifacendosi sempre all'esempio, il testo «PRODUZIONE DI VESTITI E SCARPE» sarebbe stato trasformato in «ABBIGLIAMENT CALZATUR PRODUZION».

Figura 17 - Parsing results



2.5.3 Formato dei file di output

Come già detto, il sistema di codifica produce 4 file:

- Unique.out
- Multiple.out
- Possible.out
- Fail.out.

Tutti questi file hanno lo stesso formato: sono file sequenziali i cui record sono costituiti da due sezioni. La prima i contiene il record di input (per intero) e la seconda una serie di blocchi aggiunti da Actr. La prima sezione avrà la lunghezza corrispondente al record di input, mentre i blocchi della seconda hanno lunghezza fissa. Le informazioni più salienti riportate in questi blocchi sono:

- il codice assegnato,
- il testo del database Actr con cui si è realizzato il match con il testo di input,
- il punteggio ottenuto,
- altre informazioni di sistema.

Nella tavola seguente, si riportano gli unici limiti previsti per il file da codificare, relativi ai campi di interesse per Actr.[1]

Tavola 4 - Limiti massimi dei campi di interesse per Actr del file da codificare

Nome del campo	Lunghezza	Tipo
Testo di Input	0 .. 200	C
Filtro	0 .. 10	C
Shunt	0 .. 10	C
Identificativo del record	0 .. 30	C

Le tavole seguenti mostrano invece il contenuto informativo di ciascun blocco.

Tavola 5 - Signature Block

Nome del campo	Lunghezza	Tipo
«Firma» di Actr	10	C
Versione del record	2	C
Formato del record	2	C
Riservato	4	C
Lunghezza del blocco	18	

Tavola 6 - Result Block

Nome del campo	Lunghezza	Tipo
Label del risultato	2	C
Flags del risultato	4	C
Codice assegnato	10 (massimo)	C
Rank	4	I
Punteggio	9	F
Tag del record nel Reference	10	I
Lunghezza del blocco	39	

Il Result Block è il blocco più rilevante per l'utente, in quanto contiene i risultati del match, ossia il codice ed il punteggio ottenuto.

[1] Per convenzione, d'ora in poi, il Tipo sarà identificato con:
- C → carattere
- I → intero
- F → floating point

Tavola 7 - Project Block

Nome del campo	Lunghezza	Tipo
Nome del progetto	8	C
Nome del contesto	8	C
Nome della strategia di codifica	10	C
Lunghezza del blocco	26	

Il Project block identifica il progetto utilizzato da Actr per la codifica, nonché i nomi dei componenti del progetto. Queste informazioni permettono all'utente di identificare il database e la strategia utilizzati.

Tavola 8 - Control Block

Nome del campo	Lunghezza	Tipo
Numero di sequenza del match di Input	10	I
Numero di sequenza del match di Output	4	I
Tipo di elaborazione (batch/interattiva)	1	C
Riservato	3	C
Orario	14	C
Lunghezza del blocco	32	

Il Control Block contiene l'orario dell'elaborazione, i numeri di sequenza (rispettivamente dei record di input e di output) e altre informazioni correlate.

Tavola 9 - Layout Block

Nome del campo	Lunghezza	Tipo
Posizione di questo blocco	6	I
Posizione del testo di input	6	I
Lunghezza del testo di input	6	I
Posizione del filtro di input	6	I
Lunghezza del filtro di input	6	I
Posizione dello shunt di input	6	I
Lunghezza dello shunt di input	6	I
Posizione dell'identificativo del record di input	6	I
Lunghezza dell'identificativo del record di input	6	I
Lunghezza del blocco	54	

Il Layout Block contiene le posizioni di ciascun campo del record di input di interesse per Actr (c'è zero se il campo non è previsto nel record). Questo blocco può essere utilizzato per rendere il record "auto-esplicativo" in modo da poter essere letto da altre applicazioni.

Tavola 10 - Reference Block

Nome del campo	Lunghezza	Tipo
Filtro del Reference	10	C
Testo del reference	200	C
Lunghezza del blocco	210	

Il Reference Block, infine, contiene il filtro ed il testo del *Reference* abbinati al testo di input a seguito del match.
Il demarcatore di fine linea (EOL) è l'ultimo campo del record a lunghezza fissa. La lunghezza di questo demarcatore è 1 ed il tipo "C".
I blocchi vengono aggiunti al record di input secondo un ordine fisso; ogni blocco addizionale aumenta il livello di dettaglio sul match realizzato.
Ogni blocco aggiunto a quello iniziale (Signature Block) avrà un peso in termini di spazio disco e tempi di elaborazione (la lunghezza totale di tutti i blocchi di output è di circa 380 byte). L'ordine dei blocchi non può essere alterato; è infatti stato pensato per fornire prima le informazioni più essenziali, e le altre successivamente. Ciò consente all'utente di ottimizzare le elaborazioni di grandi file, limitando l'utilizzo di spazio disco.
Sul record generato da un match fallito, oppure vuoto, oppure che ha dato adito a overflowed, i campi numerici non utilizzati in ogni blocco sono riempiti con zeri, mentre i campi carattere sono riempiti da blank.

2.5.3.1 Descrizione dettagliata dei blocchi di output

La descrizione che segue può essere utile in particolare a chi vuole utilizzare gli output d Actr tramite un'altra applicazione software, come per esempio un caricamento in un database, un foglio elettronico oppure un programma personalizzato.

Signature Block

Questo blocco viene **sempre scritto**. Inizia con una stringa che definisce la «firma» di Actr, che delimita l'inizio dei blocchi aggiunti da Actr in fase di match.
Dopo la firma, c'è una stringa di due caratteri rappresentante la versione del record; il numero di blocchi di output, la loro composizione, l'ampiezza e la relativa posizione sono impliciti nella versione del record.
Segue il record format, costituito da due caratteri.
Il primo carattere indica se il record è stato forzato ad una lunghezza fissa:

- «V» → non è presente alcun EOL; la lunghezza del record può variare. Ciò può far risparmiare spazio se vengono scritti tutti i blocchi e se i testi del *Reference* sono mediamente abbastanza brevi;
- «F» → un EOL (un carattere diverso da blank e, preferibilmente, non un alfanumerico) è posizionato in una posizione fissa del record, immediatamente dopo l'ultimo blocco alla fine del record (abitualmente si usa «*», oppure «<», eccetera).

Il secondo carattere del record format indica quali set di blocchi vengono aggiunti al record:

«0» = Soltanto il Signature Block,
«1» = Anche il Result Block (Codice, punteggio …),
«2» = Anche il Project Block (Nome del progetto, contesto e strategia),
«3» = Anche il Control Block (Sequenza delle operazioni, orario..),
«4» = Anche il Layout Block (Posizione e lunghezza dei campi di input),
«5» = Anche il Reference Block (Testo e filtro del database).

Inoltre alcuni byte sono riservati ad ulteriori flag.

Result Block

Come già detto, questo è il blocco più rilevante per l'utente, in quanto contiene i risultati del match, ossia il codice ed il punteggio ottenuto.
La label può assumere uno dei seguenti significati:

Label	**Tipo di risultato**
«U»	Unico
«M»	Multiplo
«P»	Possibile
«F»	Fallito
«E»	Vuoto
«O»	Overflow (troppi record del *Reference* generano un match)

Il primo carattere della label indica il risultato finale a livello di contesti multipli, qualora il progetto li preveda. Indica quindi in quale dei file di output il record viene scritto.
Il secondo carattere indica il risultato a livello di contesto singolo. Le due label saranno identiche ad eccezione del seguente caso: se più di un database ha generato un match *"Unico"*, il risultato finale sarà *"Multiplo"* (M), nonostante che il singolo risultato sia *"Unico"* (U).
Quando il risultato finale di un match è *"Fallito"*, *"Vuoto"*, oppure *"Overflowed"*, il risultato a livello di contesto singolo è automaticamente impostato uguale. Ciò perché non sono stati generati risultati specifici di contesto (un match *"Fallito"*, *"Vuoto"*, oppure *"Overflowed"* produrrà comunque un singolo record di output, indipendentemente da quanti contesti siano previsti nella strategia).
Seguono quattro flag di cui soltanto due sono utilizzati nella presente versione. Il primo assume valore «1» se è stato realizzato un match diretto e «0» se indiretto. Il secondo assume valore «1» se era previsto il filtro, ma è stato effettuato il «drop» e «0» se il match è stato realizzato a seguito del primo passaggio.
Il rank rappresenta la posizione nell'ambito della lista basata sul punteggio di tutti i record di output generati da un match su un testo di input. Si noti che quando le liste di output sono ordinate per punteggio, il numero di rank corrisponde all'ordine del record nella lista.

Il codice, il punteggio e la tag sono dati di cui già si è parlato.

Project Block

Tale blocco contiene i nomi del progetto, del contesto e della strategia di codifica in modo da identificare univocamente da quale ambiente di codifica il record di output sia stato generato. Soltanto sui *"Falliti"* non è riportato il nome del contesto.

Control Block

Il numero di sequenza del record di input da codificare, ogni qual volta si inizi un'applicazione di codifica (interattiva oppure batch), è *"resettato"* e viene incrementato per ogni nuovo record da codificare.

Il numero di sequenza del record di output, invece, viene *"resettato"* ogni qual volta il sistema inizia ad esaminare un record da codificare e viene incrementato per ogni nuovo record di output prodotto (nel caso degli Unici, quindi, sarà sempre uguale a «1», per i *"Multipli"* e *"Possibili"* potrà raggiungere al più il numero massimo di match da produrre in output previsto nella strategia, nel caso dei *"Falliti"* sarà sempre uguale a «0»).

Il tipo di elaborazione può assumere uno dei seguenti valori:

- «B» → batch,
- «I» → interattivo,
- «X» → tipo di elaborazione sconosciuta.

L'orario è una stringa di digit nel seguente formato: «YYYYMMDDhhmmss» (anno, mese, giorno, ora, minuti, secondi).

Layout Block

Questo blocco contiene le informazioni sulla posizione e la lunghezza di una serie di campi. I relativi valori sono uguali a zero se il campo non è presente sul record.

Il primo elemento di questo blocco riporta la posizione del blocco stesso sul record, che può essere utile per individuare eventuali cambiamenti nel record di input. L'identificativo del record di input è stato aggiunto di recente nei blocchi di Actr, ma non è pienamente supportato da tutti i programmi Actr; è stato pensato per permetterne il display da parte di Actr, qualora sia presente sul record di input. L'utilizzo di questo identificativo è comunque opzionale.

Reference Block

Il campo filtro in questo blocco è impostato a zero qualora non sia presente nel *Reference*, altrimenti viene riportato qui, indipendentemente dal fatto che sia stato utilizzato per la codifica.

Se è stata scelta l'opzione della lunghezza fissa, il campo con il testo del *Reference* è riempito di zeri fino alla sua lunghezza teorica prevista (il massimo è 200 byte); quindi viene scritto sul record un ulteriore carattere stampabile alla fine del record. Se invece è stata scelta l'opzione di

lunghezza variabile, questo carattere viene scritto subito dopo l'ultimo byte pieno del testo del *Reference*.
Una procedura simile è applicata a ogni blocco che venga eventualmente scritto come ultimo del record.

2.5.4 Ulteriori dettagli tecnici

Actr fornisce una **funzione di reporting** , attraverso un programma batch chiamato **GCDUMP**, che, fondamentalmente, effettua il dump del contenuto del database in un report che può essere stampato, oppure salvato ed editato su disco (se ne sconsiglia la stampa, viste le dimensioni).

La sintassi del comando è:

«GCDUMP *ACTR -database-filename*»

Si riporta di seguito un esempio di output prodotto da tale funzione.

```
 DATABASE CONTROL INFORMATION:
  ============================

    Software Information
    =-=-=-=-=-=-=-=-=-=-=-

       ACTR  Version:      03
       ACTR  Release:      00
       ACTR  Compat. Level: 00

    Database Information
    =-=-=-=-=-=-=-=-=-=-=-

       Database still has room to expand
       Database integrity is OK

    Context General Information
    =-=-=-=-=-=-=-=-=-=-=-=-=-=-=
       Description:           "Refs 10 Test"
       Creation date:         20-12-1996 08:49:07
       Current State:         3 (Ready to Code)
       Nbr of reference record: 10
       Last used Tag:         10
       Longest reference text:  37
```

Longest filter: 4
Longest code: 8
CPK length used: 30
Total of CPK overflow: 0
Total without overflow: 10
Parsing strategy: c:\ACTR v3\NT\DATA\PARSE\Default.GPS
Digraph & Trigraph: c:\ACTR v3\NT\DATA\GRAPH\CPKCHRE.TXT

Context Historical Information
=-=

Time Stamp	Transition	New State
20-12-1996 08:49:25	07 (Code Prep.)	03 (Ready to Code)
20-12-1996 08:49:18	04 (LOAD Ref. file)	02 (Ref. LOADed/updated)
20-12-1996 08:49:07	14 (Create database)	01 (Empty CTX; no ref. LOADed)

ORIGINAL PHRASE

TEXT: ROWID [TAG] (REFERENCE_TEXT) DCLSCNT CPKOVFL CODE FILTER _____ CPK ____

TEXT: 0 1 ACCOUNTING AGENCY 0 N 00000005 1773 026B1FFFBA2243

TEXT: 1 2 ACCOUNTING BUSINESS 0 N 00000007 1773 026B1F

TEXT: 2 3 AUTO DEALORSHIP 0 N 00000163 1631 0F1C648BFF41914F

TEXT: 3 4 AUTO ELECTRIC REPAIR 0 N 00000164 1635 41914FFFE3F5FF5B6F

TEXT: 5 6 CAR AND TRUCK DEALERSHIP 0 N 00000589 1631 9648BFF3C52FFEA9E

TEXT: 6 7 CAR AND TRUCK MANUFACTURE 0 N 00000590 1323 0FFC6CAC752FFEA9E

TEXT: 7 8 LEAD ZINC MINE 0 N 00002640 1061 ED45FF5A0643

TEXT: 8 9 TRAVEL AGENCIE 0 N 00005443 1996 BA22A7FFD456

TEXT: 9 10 TRAVEL SERVICE 0 N 00005445 1996 DAE0FFD456

WORD: ROWID WEIGHT NBUCD _____ PARSED WORD _____

WORD: 0 0.6990 2 ACOUNT

WORD: 1 1.0000 1 AGENC

WORD: 2 1.0000 1 ALORSHIP

WORD: 3 0.6990 2 AUTO

WORD: 4 1.0000 1 ELECTR

WORD: 5 1.0000 1 PAIR
WORD: 6 1.0000 1 AUTOMOBIL
WORD: 7 1.0000 1 DEAL
WORD: 8 1.0000 1 NEW
WORD: 9 1.0000 1 USED
WORD: 10 1.0000 1 VEHICL
WORD: 11 1.0000 1 ALERSHIP
WORD: 12 0.6990 2 CAR
WORD: 13 0.6990 2 TRUCK
WORD: 14 1.0000 1 MANUFACTUR
WORD: 15 1.0000 1 LEAD
WORD: 16 1.0000 1 MINE
WORD: 17 1.0000 1 ZINC
WORD: 18 1.0000 1 AGENCI
WORD: 19 0.6990 2 TRAV
WORD: 20 1.0000 1 SERVIC

LINK: ROWID TEXTID WORDID NBOCC __FILTER__
LINK: 0 0 0 1 1773
LINK: 1 0 1 1 1773
LINK: 2 1 0 1 1773
LINK: 3 2 2 1 1631
LINK: 4 2 3 1 1631
LINK: 5 3 3 1 1635
LINK: 6 3 4 1 1635
LINK: 7 3 5 1 1635
LINK: 8 4 6 1 1631
LINK: 9 4 7 1 1631
LINK: 10 4 8 1 1631
LINK: 11 4 9 1 1631
LINK: 12 4 10 1 1631
LINK: 13 5 11 1 1631
LINK: 14 5 12 1 1631
LINK: 15 5 13 1 1631
LINK: 16 6 12 1 1323
LINK: 17 6 14 1 1323
LINK: 18 6 13 1 1323
LINK: 19 7 15 1 1061
LINK: 20 7 16 1 1061
LINK: 21 7 17 1 1061
LINK: 22 8 18 1 1996

```
LINK:  23    8    19    1   1996
LINK:  24    9    20    1   1996
LINK:  25    9    19    1   1996

WDCD: ROWID  WORDID  __ CODE __  NBPHR
WDCD:   0     0   00000005     1
WDCD:   1     1   00000005     1
WDCD:   2     0   00000007     1
WDCD:   3     2   00000163     1
WDCD:   4     3   00000163     1
WDCD:   5     3   00000164     1
WDCD:   6     4   00000164     1
WDCD:   7     5   00000164     1
WDCD:   8     6   00000210     1
WDCD:   9     7   00000210     1
WDCD:  10     8   00000210     1
WDCD:  11     9   00000210     1
WDCD:  12    10   00000210     1
WDCD:  13    11   00000589     1
WDCD:  14    12   00000589     1
WDCD:  15    13   00000589     1
WDCD:  16    12   00000590     1
WDCD:  17    14   00000590     1
WDCD:  18    13   00000590     1
WDCD:  19    15   00002640     1
WDCD:  20    16   00002640     1
WDCD:  21    17   00002640     1
WDCD:  22    18   00005443     1
WDCD:  23    19   00005443     1
WDCD:  24    20   00005445     1
WDCD:  25    19   00005445     1
```

Relativamente, infine, alla stima dello spazio disco necessario per il database, come già detto, l'allocazione di default effettuata da Actr è di un megabyte, anche se venissero caricati soltanto due record.

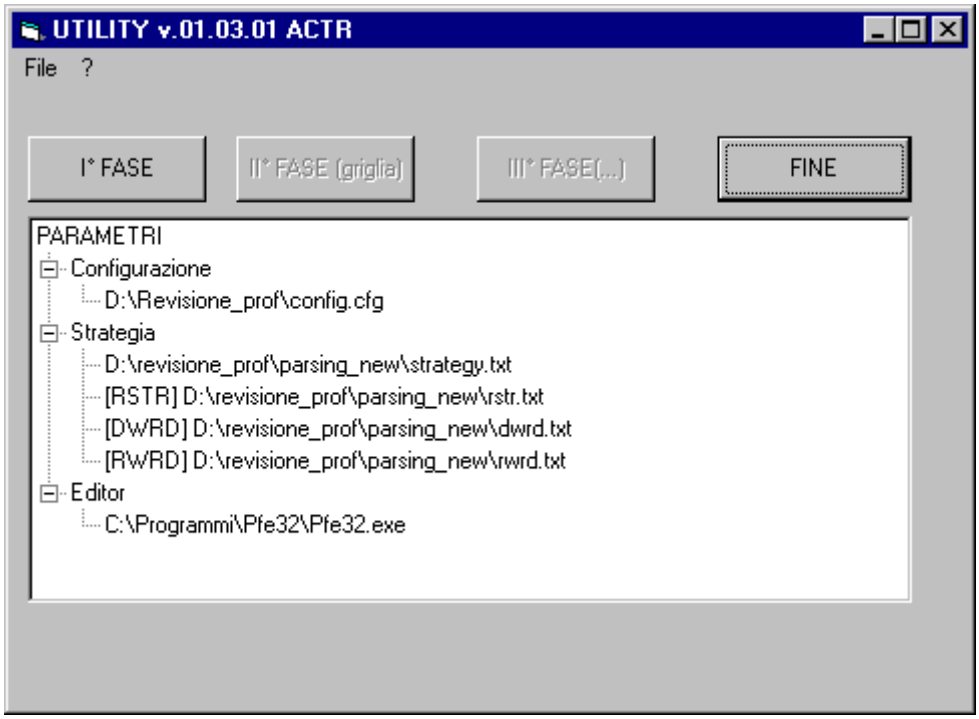
Man mano che il database cresce, questo si espande poi automaticamente, di segmenti di 0.5 MB (512 Kb).

Molto approssimativamente, il database creato da un *Reference file* sarà di una dimensione 5 volte più grande del *Reference file*, in termini di byte; per esempio, un *Reference* di 300 MG genererà un database di 1.5 GB. Ciò che incide sulla dimensione finale è la lunghezza del testo e l'eventuale presenza del filtro (l'esempio citato considera un testo di 200 byte).

3 Software di supporto alla gestione del sistema

Come già esposto nei capitoli precedenti (confrontare paragrafo 2.4.1), il sistema Actr, al momento del caricamento del database, effettua un controllo finalizzato a scartare i record *“non validi”*, ossia quelli che, a seguito del parsing:

-



La **prima fase,** che viene lanciata cliccando sull'apposito bottone "I FASE", effettua una serie di controlli singolarmente su ciascun file; i passaggi realizzati sono i seguenti:

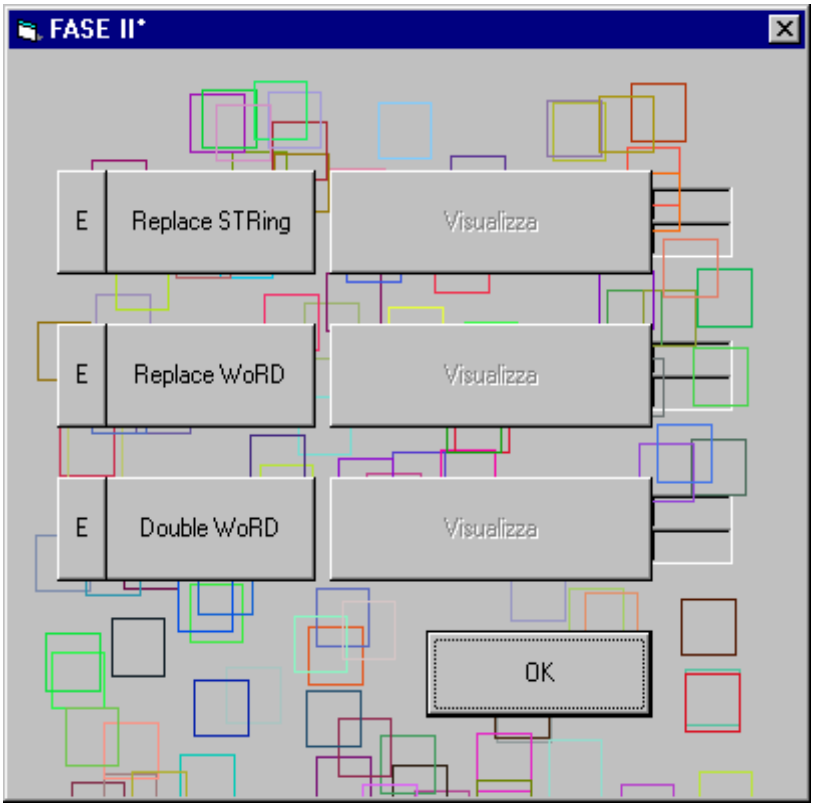
- ordinamento del file, secondo i seguenti criteri:
  RSTR → ordinamento alfabetico sulla seconda colonna
  RWRD → ordinamento alfabetico sulla seconda colonna, quindi sulla terza
  DWRD → ordinamento alfabetico sulla terza colonna, quindi sulla quarta
- eliminazione delle schede duplicate
- eliminazione delle schede nelle quali i termini delle colonne di destra sono uguali a quelli delle colonne di sinistra (trasformazione A → A)

La **seconda fase**, invece, è finalizzata a mettere in luce diversi tipi di incoerenze all'interno di ciascuno dei file citati ed a memorizzarle su appositi file Excel. In particolare si procede:

- all'individuazione delle trasformazioni incoerenti, quali ad esempio:
  trasformazione di A → B
  trasformazione A → C
- all'individuazione delle trasformazioni ricorsive, quali ad esempio:
  trasformazione di A → B
  trasformazione di B → C
  oppure
  trasformazione di A → Z B
  trasformazione di B → C D

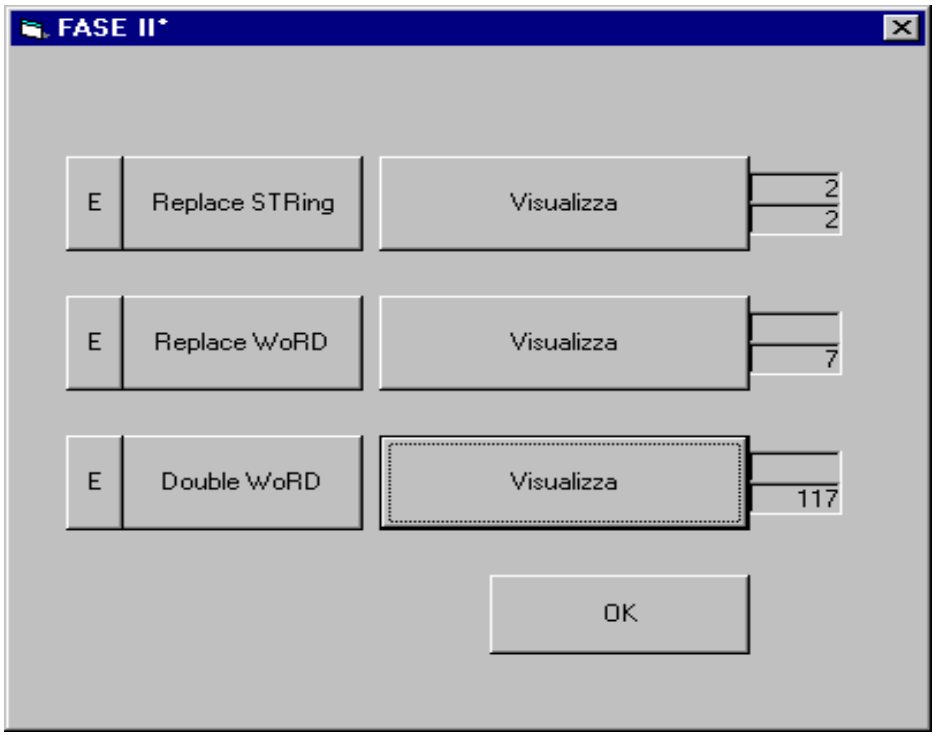
Come può vedersi dalla schermata, di cui in figura 19, viene richiesta l'esecuzione della procedura su ciascuno dei tre file in sequenza; le incoerenze sono registrate in appositi file Excel e viene inoltre data la possibilità di correggere agilmente i file originari, cliccando sul bottone "E" che consente di editare immediatamente ciascun file implicato nel controllo.

Figura 19 - Controlli di coerenza del parsing: seconda fase



Al termine di ciascun passaggio, come può vedersi dalla figura 20, non soltanto è evidenziato il bottone "VISUALIZZA", che consente di editare i file Excel, ma sono valorizzati una serie di contatori, nei quali sono riportati i totali delle schede da correggere.

Figura 20: **Controlli di coerenza del parsing: seconda fase a conclusione dei passaggi**



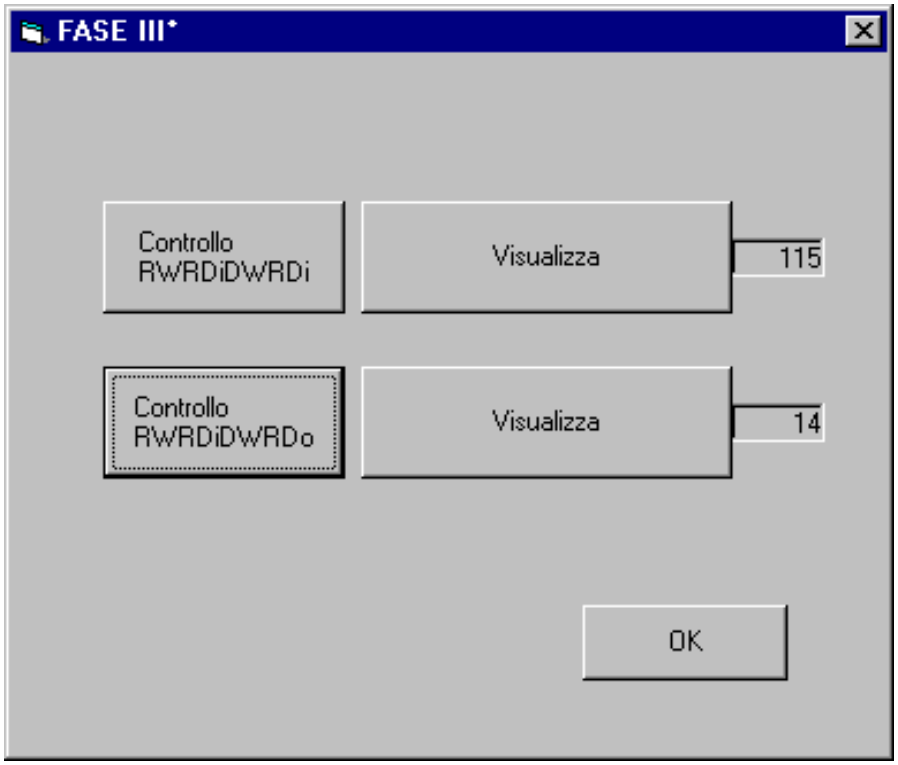
Con la **terza fase**, infine, vengono realizzati i controlli tra il file delle RWRD e quello delle DWRD.

In particolare si procede:

- all'individuazione di coincidenze tra la prima colonna del file RWRD e le prime due colonne del file DWRD, ad esempio:
  trasformazione in RWRD di A → Z
  quindi trasformazione in DWRD di A B → C D
  oppure trasformazione in DWRD di B A → C D
- all'individuazione di coincidenze tra la prima colonna del file RWRD e le seconde due colonne del file DWRD, quali ad esempio:
  trasformazione in RWRD di A → B
  quindi trasformazione in DWRD di C D → A B
  oppure trasformazione in DWRD di C D → B A

Si fa presente, comunque, che quest'ultimo tipo di coincidenze non costituisce necessariamente un errore, quindi non è indispensabile un intervento correttivo, ma può essere sufficiente verificare che non si tratti di una svista, ma di una trasformazione voluta.

Figura 21 - Controlli di coerenza del parsing: terza fase



L'utilizzo corretto di questa applicazione implica che, ogni qual volta si effettui un intervento correttivo in un qualsiasi passaggio di ciascuna fase, sia opportuno eseguire nuovamente la procedura dall'inizio, in modo da evidenziare le effettive schede incoerenti corrispondenti all'ultima versione di ciascun file di parsing.

4 Actr v3: Guida all'utilizzo di Actr

Descrizione del metodo utilizzato per la standardizzazione linguistica della Classificazione delle Attività economiche[1] allo scopo di applicare la codifica automatica con il sistema Actr

Standardizzazione linguistica

Come già descritto nella prima parte di questo manuale, Actr prevede una fase di standardizzazione linguistica definita "*parsing"*, sulla base della quale si rende possibile il *matching* tra le descrizioni-risposta e le descrizioni contenute nel dizionario processabile. Per quanto riguarda le funzioni e la strategia di *parsing* si rimanda perciò, alla esposizione che ne viene fatta, nel paragrafo 2.3.[2] In questa sezione, ci si limita più semplicemente a descrivere gli accorgimenti pratici e la tecnica di compilazione dei processi, così come sono stati utilizzati durante l'implementazione dell'ambiente di codifica di Actr sulla Classificazione delle attività economiche 1991.
Si precisa che, in questa applicazione, l'ordine con cui vengono eseguiti i processi è coerente con la figura 2 di questo manuale.
Da un punto di vista pratico, le operazioni di standardizzazione del testo, dirette a ridurre la variabilità linguistica, sia delle descrizioni contenute nel dizionario processabile (*Reference file*), sia delle descrizioni-risposta (*Input file*) da codificare, possono essere ricondotte a due tipi di intervento che potremmo definire di standardizzazione grammaticale e di standardizzazione dei significati.
Infatti, Actr prevede delle funzioni specifiche (*parsing*) per la standardizzazione degli aspetti grammaticali del testo, di quelli morfologici (le parti del discorso), di quelli fonologici (la punteggiatura). Queste stesse funzioni di *parsing* permettono inoltre la standardizzazione dei significati riguardo alla assegnazione di significati uniformi alle parole e alla definizione di classi di concetti valide contestualmente.
Entrambi gli interventi sono a carico dell'utente e, di fatto, consistono nella compilazione, secondo formati stabiliti, di alcuni *data file*[3]*di parsing*, contenuti negli step di *Phrase Processing* e di *Word Processing*.
I primi, relativi alla standardizzazione grammaticale, agendo su tutto il testo, sono scelti all'inizio della strategia e non richiedono frequenti cambiamenti. Gli altri, relativi alla standardizzazione dei significati, agendo su parole specifiche sia del dizionario processabile, sia del testo da codificare, richiedono continue operazioni di compilazione, controllo e verifica dei file dati relativi ai processi.

[1] Istat. *Classificazione delle attività economiche* Roma: Istat 1991 (Metodi e norme, serie C-11).
[2] Per tutti gli aspetti informatici si rimanda alla prima parte dove sono trattati in maniera specifica nella "Guida al Parsing".
[3] Per i formati di tutti i file di parsing si rimanda alla prima parte. In particolare alla lettura del paragrafo 2.3.4, dove sono trattati in modo specifico.

4.1 Standardizzazione morfologica-grammaticale

Nell'ambito dell'applicazione del software Actr al testo della Classificazione delle attività economiche, le parti del discorso,[4] prese in considerazione ai fini della standardizzazione, sono gli articoli, le preposizioni semplici e articolate, le congiunzioni, i sostantivi e gli aggettivi. Infatti, in questo specifico contesto, non si incontrano né le forme verbali, né i pronomi, né tantomeno le interiezioni. La variabilità morfologica del discorso osservata nelle descrizioni dell'attività economica si limita alla declinazione degli articoli, delle preposizioni articolate, dei sostantivi e degli aggettivi, cioè alle variazioni di desinenza secondo il genere e il numero. In realtà gli aggettivi variano anche secondo il grado[5] e i sostantivi anche secondo la loro modalità di formazione.[6] Però, in questo contesto, gli aggettivi non presentano tutta la variabilità, in quanto sono sempre usati al grado positivo, (del resto, molti aggettivi qualificativi indicanti attributi della materia esempio: "argenteo", "sulfureo", eccetera, in italiano non ammettono altri gradi). Invece, per quanto riguarda i sostantivi, questi sono usati in tutte le forme ad eccezione di quella alterata. Quindi, la variabilità rappresentata dai sostantivi è determinata dalla declinazione del genere e del numero relativa al tipo specifico di nome primitivo, derivato o composto.

4.1.1 Standardizzazione di articoli, preposizioni, congiunzioni e avverbi

In questa applicazione di codifica automatica (così come nella gran parte di quelle sviluppate), sono escluse dal testo tutte le parti invariabili del discorso, le preposizioni semplici, le congiunzioni e gli avverbi, perché si ritiene che siano elementi di ostacolo alla uniformità delle descrizioni. Anche, gli articoli e le preposizioni articolate sono esclusi nello stesso modo, perché sono considerati elementi che diversificano, in maniera non significativa, la struttura di descrizioni altrimenti simili. La cancellazione serve a evitare che descrizioni ottenute dalla combinazione dei medesimi sostantivi e aggettivi, risultino diverse tra loro per la presenza nel testo di preposizioni, articoli e congiunzioni. Così facendo, depurate di tutte le parti del discorso diverse da sostantivi e aggettivi risultano eguali a un medesimo elenco di parole.
Gran parte della cancellazione di questi elementi avviene con il processo *Replacement Words* (il quale, come viene spiegato nei paragrafi 4.5, 4.5.1, 4.5.2, non è utilizzato esclusivamente a questo

[4] Tutte le parole di cui noi possiamo servirci, nello scrivere e nel parlare, si suddividono in nove distinte classi chiamate parti del discorso. Di queste nove parti del discorso, cinque si dicono variabili, perché possono variare, cioè modificare la forma secondo il numero e il genere, o, se si tratta del verbo, secondo la persona, il tempo, il modo e la forma. Le altre quattro, che sono dette invariabili, non vanno mai soggette ad alcun mutamento. Sono variabili: il nome, l'articolo, l'aggettivo, il pronome, il verbo. Sono invariabili l'avverbio, la preposizione, la congiunzione e l'interiezione. Fra le parti variabili, il nome, l'articolo, l'aggettivo e il pronome si declinano, cioè possono variare la desinenza secondo il genere (maschile e femminile) e il numero (singolare e plurale). Il verbo, invece, si coniuga ossia va soggetto a moltissimi mutamenti, non solo rispetto al genere e al numero, ma anche rispetto alla persona, al tempo, al modo, alla forma

[5] L'aggettivo qualificativo può esprimere la qualità in tre gradi diversi:
1) positivo, se la esprime in modo semplice;
2) comparativo, se nell'esprimerla stabilisce anche un confronto di maggioranza, di minoranza, di eguaglianza;
3) superlativo, se la esprime nel suo grado massimo, sia in senso assoluto che relativo.

[6] A seconda della formazione i sostantivi possono essere primitivi, alterati, derivati e composti. I nomi primitivi sono costituiti di tema e di desinenza. I nomi alterati si ottengono mettendo al posto della desinenza un appropriato suffisso accrescitivo, diminutivo, vezzeggiativo, peggiorativo. Le parole derivate si ottengono applicando alla radice della parola originaria prefissi e suffissi. Vanno distinti dai derivati i nomi composti i quali sono formati dall'unione di due parole.

scopo, ma trova largo impiego in altri tipi di sostituzione). In pratica, è sufficiente trascrivere nel file dati relativo al processo *Replacement Words (Rwrd)* le preposizioni, le congiunzioni, gli avverbi, gli articoli che si ritiene vadano cancellati per ridurre il testo a un semplice elenco di parole.

Esempio:

a) Fabbricazione (di)[7] infissi (in) legno.
a') Produzione (di) porte (e) finestre (di) legno.

Nel caso degli esempi vengono eliminate le due preposizioni semplici "di" e "in" e la congiunzione "e". In questo modo si ottengono due liste di parole: la prima costituta da "fabbricazione infissi legno", la seconda costituita da "produzione porte finestre legno". Ovviamente, si sarebbe ottenuto lo stesso risultato, se invece di preposizioni e congiunzioni si fosse trattato di articoli ("il", "lo", ...) o di avverbi o di altre parti del discorso ritenute non significative. Per questo motivo, possono essere eliminati anche alcuni aggettivi che, si ritiene, non aggiungano significato ai fini della corretta individuazione della categoria di attività economica; ma si ritiene che, piuttosto, introducano un fattore di variabilità nella descrizione di una stessa attività e ostacolino la messa a punto di descrizioni standard. È questo il caso di parole come "similare", "simile", "qualsiasi", "ogni".

- Alcuni accorgimenti necessari per la standardizzazione di "in" e di "ed".[8]

In alcune descrizioni la preposizione semplice "in" non può essere esclusa dal testo, a meno di incorrere in confusioni di significato e in errori di attribuzione del codice. Ad esempio, le due espressioni "produzione di abbigliamento in stoffa" e "produzione di stoffa per abbigliamento", in base alle sostituzioni effettuate in modo generalizzato dal processo *Rwrd*, risulterebbero uguali, mentre, al contrario, implicano (oltre alle ovvie differenze di significato) attività e codici differenti.
In casi come questo, si può trascrivere nel file dati relativo al processo *Rstr* l'espressione "in stoffa" sostituendola con un'unica stringa "INSTOFFA". La preposizione "in", limitatamente a questa descrizione, è sottratta all'esclusione che avviene in *Rwrd* e l'attività può essere codificata in modo corretto.
Un caso analogo è rappresentato dalla congiunzione "ed". In alcune descrizioni la parola "ed" è usata come forma abbreviata di "edilizia" e di "edile". Ad esempio, può accadere che l'espressione "lavoro edile" sia scritta, per brevità, "lavoro ed.". È evidente che l'abbreviazione, sebbene impropria, risulta indispensabile all'individuazione dell'attività svolta. Perciò è necessario trovare un accorgimento per fissarne il significato contestuale. Come nel caso precedente, si ricorre al processo *Rstr*.
In particolare, "ed" viene considerata una congiunzione se seguita da parole che iniziano per vocale; quindi nel file dati *Rstr* viene trasformata in "e". Le schede della Rstr sono le seguenti:

ED A	*(diventa)* E A
ED E	*(diventa)* E E
ED I	*(diventa)* E I
ED O	*(diventa)* E O
ED U	*(diventa)* E U

[7] Le lettere e le parti grammaticali incluse tra parentesi sono da considerarsi eliminate.
[8] Macchia Stefania. *Sperimentazione, implementazione e gestione dell'ambiente di codifica automatica della Classificazione delle attività economiche.* Roma: Istat, 2002 (Documenti Istat, n. 2/2002).

In questa fase, “ed”, se seguita da parole che iniziano per consonante, viene quindi lasciata intatta. Successivamente, “e” viene sostituita con Blank nel processo *Rwrd*, mentre “ed” viene trattata nel processo *Dwrd* a seconda delle parole che la precedono o la seguono, per esempio:

LAVORAZIONE ED *(diventa)* LAVORAZIONE EDILIZIA

- Un caso particolare: la congiunzione “non”.

Un caso particolare è rappresentato dalla congiunzione “non” in quanto non è soggetta a eliminazione. Questo non solo per evidenti ragioni di significato, ma perché, nel contesto considerato, le classi di attività economica sono spesso definite per negazione. È frequente che l’appartenenza ad una certa classe sia espressa come esclusione da un’altra: dall’assenza, piuttosto che dalla presenza di determinate caratteristiche. Questo tipo di descrizioni è reso più frequentemente con espressioni formate da “non + *aggettivo*” o da “senza + *sostantivo*”.
In pratica nel file dati *Rwrd* i termini indicanti negazione sono tutti uniformati alla congiunzione “non”.

Esempio:

ECCETTO	*(diventa)* NON
ECCEZIONE	*(diventa)* NON
ESCLUSO	*(diventa)* NON
ESCLUSI	*(diventa)* NON
ESCLUSA	*(diventa)* NON
ESCLUSE	*(diventa)* NON
ESCLUSIONE	*(diventa)* NON
EXTRA	*(diventa)* NON
NO	*(diventa)* NON
SENZA	*(diventa)* NON

Successivamente, nel file dati *Dwrd* (confrontare paragrafi 4.7, 4.7.1, 4.7.2) le coppie di parole costituite da “non + *aggettivo*” e da “non + *sostantivo*” sono sostituite da una “parola unica” nella quale la congiunzione “non” è direttamente attaccata alla stregua di un prefisso.

Esempio:

NON SPECIALIZZATO	*(diventa)* NONSPECIALIZZATO
NON GOMMA	*(diventa)* NONGOMMA.

Le parole negative, ottenute con i prefissi privativi “de” e “a”, sono sostituite con una parola unica nella quale la congiunzione “non” prende il posto del prefisso.

Esempio:

DESPECIALIZZATO *(diventa)* NONSPECIALIZZATO

La creazione di questo tipo di “parole uniche” ha lo scopo di vincolare la posizione della congiunzione “non” alla parola cui si riferisce, prima che sia eseguito il s*ort*. Infatti nel s*ort*, le parole, venendo disposte in ordine alfabetico, perdono il contenuto semantico stabilito dalle loro posizioni. In particolare, la congiunzione “non” potrebbe essere attribuita a qualunque altro termine. Mentre il termine cui si riferisce potrebbe acquistare un significato affermativo esattamente opposto

a quello della descrizione nella quale è contenuto. Questo significa che nel processo di *matching* potrebbero essere abbinate, con una certa probabilità, proprio a quelle categorie di attività che sono all'opposto di quelle descritte in modo negativo. Ad esempio, l'espressione "non specializzato", venendo scissa in due distinte parole indipendenti, potrebbe dare luogo tanto ad abbinamenti con descrizioni nelle quali sia compreso "non" riferito a "specializzato", quanto con descrizioni che comprendono la parola "non" riferita a un termine diverso da "specializzato".
Prendiamo ad esempio la descrizione:

"Commercio all'ingrosso **non** specializzato di prodotti alimentari."

Dopo il s*ort* si ottiene un elenco in cui le parole sono tutte disposte in ordine alfabetico e "non" ha perso la sua posizione accanto a "specializzato":

"Alimentar(i) Commerc(io) Ingross(o) **Non** Prodott(i) Specializzat(o)".

In questo modo è andata persa la certezza del significato: "non" può riferirsi a qualunque termine. Questo, nella pratica, potrebbe portare a classificare un'attività descritta come "non specializzata" con il codice di un'attività analoga, descritta viceversa come specializzata, creando un grave errore nel processo di codifica automatica. Con la creazione della "parola unica" "NONSPECIALIZZATO" si restringe il numero delle combinazioni di parole a quelle di maggiore senso contestuale. L'elenco, pur essendo semplicemente alfabetico, acquista un significato meno ambiguo e orienta più precisamente la codifica automatica:

"Alimentar(i) Commerc(io) Ingross(o) **Nonspecializzat(o)** Prodott(i) ".

In realtà, il risultato del *matching* è molto più complesso di un semplice abbinamento e, anche se avviene in modo automatico, risente dell'interazione di diversi fattori sia qualitativi, sia quantitativi. Infatti, si deve considerare che il *matching* avviene sulle parole già trasformate. Le parole giungono al s*ort*, dopo la standardizzazione, sistematicamente rielaborate: modificate nella forma e spesso ridefinite nel contenuto. Inoltre, si deve considerare che il *matching* è influenzato dalla differenza tra il numero di parole contenute in ciascuna descrizione-risposta da codificare e quello delle parole contenute nelle descrizioni-dizionario che codificano, dalla numerosità delle descrizioni e delle occorrenze di ciascuna parola presenti nel *Reference file*, dai valori assegnati alle soglie. Per quanto riguarda una più ampia illustrazione dell'argomento si rimanda alla prima parte, al paragrafo 2.3. In particolare, per i problemi di definizione delle soglie si rimanda a quanto viene detto nel paragrafo 2.5 sulla codifica automatica.

- Il caso di più negazioni.

Un altro possibile impiego di questo tipo di "parole uniche" negative, è quello di evitare la *Remove Duplicates* o rimozione dei duplicati (confrontare il paragrafo 2.3.2.2), quando in una descrizione ci siano due "non" significanti correttamente due diverse qualità negative, perciò non assimilabili.
Prendiamo a esempio la descrizione (COD ATECO 52.12):

"Commercio al dettaglio in esercizi **non** specializzati con prevalenza di prodotti **non** alimentari.".

Se la rimozione dei duplicati avesse luogo e se i due “non” fossero ridotti a uno solo, la descrizione del precedente esempio potrebbe essere abbinata, in modo parziale, a un’altra simile. Ad esempio potrebbe essere abbinata, anche se con un basso punteggio, alla descrizione (COD ATECO 51.38):

“Commercio all’ingrosso **non** specializzato di prodotti alimentari”.

4.1.2 Standardizzazione del genere e del numero nei sostantivi e negli aggettivi

Sono eliminate le vocali finali e alcuni gruppi vocalici e semivocalici con i quali le parole possono terminare.
Questa modifica riguarda solo le parole maggiori di quattro lettere, o che comunque, una volta eliminate le vocali o gruppi vocalici, rimangono di almeno quattro lettere, in quanto, si assume che, la minima unità linguistica di significato compiuto sia composta da non meno di quattro lettere.
In pratica, nel file dati relativo al processo definito *“Suffixes” (Sufx)* sono trascritte le vocali e i gruppi vocalici e semivocalici da eliminare. La funzione Sufx, per ciascuna parola, dapprima cerca e, se li trova, elimina i gruppi di lettere e poi, se non li trova, cerca e elimina le lettere singole. (ad esempio nella parola “camicia” toglie il gruppo “ia” e non la vocale “a”). Per maggiore chiarezza si riporta l'elenco delle lettere e gruppi di lettere finali soggette a eliminazione e attualmente trascritte nella prima e unica colonna (colonna 1-30) del file dati *Sufx*:

“A, E, EA, EE, I, IA, IE, II, IO, HI, HE, O, OA, OE, OI”

L'eliminazione della vocale e dei gruppi vocalici e semivocalici finali, privando ciascuna parola della parte terminale e privando in particolare i sostantivi e gli aggettivi del genere (maschile, femminile) e del numero (singolare, plurale), ha lo scopo di ridurli a una identica unità di scrittura, a un grafema invariante.

Esempio: aa)
Fabbricazion(e) (di) infiss(i) (in) legn(o).
Esempio: a'a)
Produzion(e) (di) porte (e) finestr(e) (di) legn(o).

Come si vede dagli esempi, dopo questa trasformazione non c'è più differenza tra il plurale e il singolare dei sostantivi in quanto si ottengono delle unità di scrittura invarianti (ad eccezione di “porte” di cui si dirà immediatamente dopo nel paragrafo 4.1.3). In questo modo le due liste di parole diventano la prima “fabbricazion infiss legn” e la seconda “produzion porte finestr legn”. Questo significa che, ad esempio, le parole “infissi” e “infisso” sono entrambe riconosciute come identiche a “infiss” e dunque identiche tra loro.

- Prefissi e suffissi.

Dal momento che, i prefissi e i suffissi[9] hanno un significato proprio che mantengono inalterato e vivo in tutte le parole derivate di cui fanno parte (ad esempio: s- e- es- = fuori da; pre- = prima; -to, -ta = nomi o aggettivi derivanti dal participio passato dei verbi; -zione, -gione, -tura, -itura = attività derivata da verbi; etc., …) non vengono eliminati ai fini della standardizzazione delle descrizioni delle attività economiche, in quanto per la varietà di questo contesto sono considerati degli indispensabili stabilizzatori semantici. Ad esempio: se venissero eliminati i suffissi -to, e -zione, rispettivamente dalle parole "lavora**to**" e "lavora**zione**", si otterrebbe per entrambe lo stesso tema "lavora-". È evidente che "lavora-", ai fini di una corretta codifica, non rappresenta una standardizzazione efficace in quanto, creando una aggregazione di significati impropria, produce una informazione generica, addirittura più carente di quella di partenza. Perciò, anche le parole derivate, che terminano con un suffisso, sono soggette, come tutte le altre, alla regola di eliminazione della vocale (o gruppo vocalico) finale.[10]

- Il caso dei nomi composti.

Questo vale anche per i nomi composti (esempio: autostrada, ferrovia, parabrezza, portalettere). Per i nomi composti, che formano il plurale in modo irregolare, è preferibile, utilizzando il file dati *Rwrd*, stabilire un'assegnazione a favore del singolare. Infatti, quando il plurale si forma aggiungendo la desinenza alla prima e non alla seconda parola del nome composto (oppure alla prima e alla seconda parola) la variazione riguarda una lettera interna al nome, perciò singolare e plurale rappresentano, ai fini del *matching*, due parole diverse, non assimilabili con la perdita della sola vocale finale.

Esempio:

CAPISALA *(diventa)* CAPOSALA
CAPICUOCO *(diventa)* CAPOCUOCO
POMIDORI *(diventa)* POMODORO
POMIDORO *(diventa)* POMODORO.

[9] I prefissi più comuni sono: **equi-**= uguale (esempio: equivalente); **moto-**= abbreviazione di motore (esempio: motocicletta); **multi-**= che ha molti (esempio: multiproprietà); **semi-**= metà (semilavorato); **fono-**= suono (esempio: fonografo); **demo-**= popolo (esempio: demografia); **foto-**= luce (esempio: fotografia); **ippo-**= cavallo (esempio: ippodromo); **termo-**= caldo (esempio: termosifone). I suffissi più frequenti sono:**-tore, -sore** che indicano agente derivato da verbi (es.: tornitore, incisore); **-ino, -ale, -oso, -esco** che servono a formare gli aggettivi (esempio: alpino, naturale, oleoso, pescoso); **-iere, -aio, -aiolo, -ario** che indicano chi esercita un'attività (esempio: barbiere, operaio, pizzaiolo, funzionario); **-ista**= agente non derivato da verbi (esempio: artista, autista, dentista); **-aio, -ile, -orio, -oio** che indicano luogo con derminata funzione, luogo d'un lavoro o mezzo con cui questo si compie (esempio: pollaio, canile, laboratorio, filatoio, rasoio); **-to, -ta** che servono a formare nomi o aggettivi con il participio passato di verbi (esempio: deposito, raccolta); **-zione, -gione, -tura, -itura** che indicano attività derivata dai verbi (esempio: lavorazione, segagione, alesatura, mungitura); **-fero** che indica la proprietà di portare o produrre (esempio: metallifero, fruttifero); **-ficio** che indica fabbricazione o luogo di fabbricazione (esempio: opificio, maglificio); **-ezza, -ità, -aggine, -ìa** che indicano nomi astratti derivati da aggettivi (sicurezza, densità, voraggine, lotteria).

[10] Questa scelta, come già detto, si giustifica in termini di opportunità e non di astratta coerenza grammaticale. Deriva dalle caratteristiche del contesto viste in funzione di un efficace *matching*. Per gli stessi motivi di opportunità, nell'ambito della standardizzazione del dizionario delle professioni, è stato seguito un altro criterio. Viceversa, è stato fatta la scelta di eliminare i suffissi che indicano l'agente di un'azione (**-tore, -sore, -ista,** etc.) perché, in questo contesto, variano sistematicamente rispetto al genere (esempio: tornitore, tornitrice).

4.1.3 Eccezioni alla regola di eliminazione delle vocali finali

All'interno di Actr è stato previsto un processo, definito *"Exception Words" (Excp)*, nel quale possono essere indicate le parole maggiori di quattro lettere che non devono sottostare alla regola generale di eliminazione delle vocali finali. Si deve precisare che questa funzione permette di intervenire solo su singole parole e non su vocali o gruppi vocalici particolari (confrontare paragrafo 2.3.2.1). Le parole che, si ritiene, debbano conservare integralmente la forma, devono essere trascritte in *Excp*. In questo modo il matching avviene sulla parola completa.
Come si sarà osservato, la seconda descrizione dell'esempio precedente (a'a) contiene una parola, "porte", completa di vocale terminale, anche se maggiore di quattro lettere. È questo proprio uno dei casi nei quali la parola viene interamente mantenuta. Più precisamente si tratta delle parole che costituiscono "falsi cambiamenti di genere", parole che si comportano contestualmente come questi e "parole primitive".

- Parole che costituiscono "falsi cambiamenti di genere".

Siamo di fronte a parole che, a seconda della desinenza, cambiano completamente significato: è evidente, ad esempio, che "porta" non è il femminile di "porto". Non si tratta, infatti, di "nomi mobili", i quali hanno desinenze diverse per il maschile e per il femminile (ad esempio, "signore" e "signora", "parrucchiere" e "parrucchiera") si tratta piuttosto di "falsi cambiamenti di genere".
In pratica, nel file dati *Ecxp* vengono trascritte delle sostituzioni, nelle quali, in prima colonna (colonna 1-30), compaiono il singolare e il plurale e in seconda colonna (colonna 31-60), il singolare della parola stessa (confrontare paragrafo 2.3.4.).
Esempio:

PORTA *(diventa)* PORTA[11]
PORTE *(diventa)* PORTA

PORTO *(diventa)* PORTO
PORTI *(diventa)* PORTO.

Un metodo leggermente diverso è quello di stabilire la sostituzione della forma plurale con quella singolare nel file dati *Rwrd* e di trascrivere nel file dati *Excp* solo la sostituzione della parola con se stessa. Questo secondo metodo non è semplicemente alternativo, ma obbligatorio, quando la parola da trattare compare nella *Dwrd*, perché la funzione *Excp* viene eseguita dopo la *Dwrd*. (Come si spiegherà meglio nel paragrafo 4.5, tutte le parole che compaiono nel file dati *Dwrd*, vanno prima sostituite con il loro singolare nel file dati *Rwrd*.).

Esempio:

date le sostituzioni nel file dati Rwrd:

PORTE *(diventa)* PORTA[12] *e* PORTI *(diventa)* PORTO,

[11] La parola "porta/e", in realtà non figura nell'attuale file dati *Exception Words*. Come si vedrà in seguito, nel paragrafo 4.3.1, viene risolta nel file dati *Rwrd*, a mezzo dell'inclusione nella classe "infisso". Questa soluzione è sembrata contestualmente più efficace. L'esempio è tuttavia valido e viene mantenuto per coerenza della sequenza esplicativa.
[12] Idem.

allora sarà nel file dati Excp:

PORTA *(diventa)* PORTA *e* PORTO *(diventa)* PORTO.

- Parole che si comportano contestualmente come "falsi cambiamenti di genere"

Sono inoltre trascritte nel file *Excp* le parole che vanno trattate in maniera analoga ai falsi cambiamenti di genere, in quanto contestualmente assumono significati diversi in base alla desinenza, a seconda che questa sia plurale o singolare, maschile o femminile. Appartiene a questo gruppo, ad esempio la parola "gomme".
Per quanto riguarda la parola "gomm**e**", la desinenza plurale è mantenuta, attraverso la trascrizione in *Excp*, perché individua svariati settori di produzione: da quello delle gomme da masticare (codice Ateco 15.84.0) a quello dei pneumatici (codice Ateco 25.11.0). Mentre il singolare "gomm**a**" è usato per indicare la fabbricazione di gomma sintetica in forme primarie e degli articoli in gomma in genere.
Per questo stesso motivo non è stato possibile costruire una classe unica contenente sia le gomme che i pneumatici. Infatti entrambe le sostituzioni risultano errate ai fini della codifica automatica: la parola "gomme" non può sostituire la parola "pneumatici" (PNEUMATICI *(diventa)* GOMME) perché identifica un aggregato eccessivamente grande ed eterogeneo rispetto ai codici dell'attività economica; viceversa, la parola "pneumatici" non può sostituire la parola "gomme" (GOMME *(diventa)* PNEUMATICI) perché individua un aggregato più piccolo, tra l'altro, precisamente individuato da uno specifico codice Ateco.

Esempio:

GOMME *(diventa)* GOMME
GOMMA *(diventa)* GOMM(A)

Come si vede dagli esempi, ci possono essere dei casi, nei quali, a differenza del solito, la sostituzione in *Excp* avviene di necessità a favore del plurale. Questo significa che le parole sottoposte a questo tipo di trasformazione (conservate al plurale), devono essere trascritte, a differenza di tutte le altre al plurale anche nel file dati *Dwrd*. (come già detto, la funzione *Dwrd* è trattata al punto 4.7 e seguenti.).

- Parole "primitive".

Un altro tipo di parole che, talvolta richiede di ricorrere alla funzione *Excp* è quello delle "parole primitive". Per maggiore chiarezza, ricordiamo brevemente che si dicono "primitive" le parole formate solamente di radice e desinenza. Quelle invece che a questi elementi aggiungono una lettera o un gruppo di lettere sono "parole derivate": le lettere aggiunte prima della radice si chiamano prefissi, quelle aggiunte dopo si chiamano suffissi.
Se ad una parola togliamo la desinenza otteniamo il tema. Ma, mentre nelle "derivate" esso è costituito dalla radice preceduta dai prefissi e seguita dai suffissi (esempio: "farmac-ist-a"), nelle "primitive" il tema è la stessa radice (esempio: "farmac-o" e "farmac-ia"). Questo significa che la perdita della desinenza abolisce ogni differenza nelle parole primitive formate da una stessa radice (esempio: "farmac").
Dal momento che da una stessa radice si ottengono, tramite la desinenza, parole di significato diverso, quando queste parole individuano diverse categorie di attività economica, è necessario rappresentare la differenza di significato in modo da raggiungere una corretta codifica. Ad esempio,

nella famiglia di parole “farmaco”, “farmacia”, “farmacista” la perdita della desinenza[13] nelle prime due produce la radice “farmac”, mentre nella terza produce il tema “farmacist”.

Esempio: eliminazione della desinenza

farmac(o)
farmac(ia)
farmacist(a).

Come si vede dall'esempio, le prime due parole aventi significato diverso sono rese irriconoscibili dalla perdita della desinenza.
Per ovviare a questo effetto la parola “farmaco” può essere trascritta nel file dati *Excp*. Mentre le altre due parole sono normalmente sottoposte alla regola generale di eliminazione della vocale finale in quanto non c'è possibilità di confonderle.

Esempio: FARMACO *(diventa)* FARMACO

In conclusione si può dire che eventuali effetti incongruenti dovuti alla perdita della desinenza sono generalmente risolti, alternativamente a seconda delle opportunità, con la funzione *Excp* o con la funzione *Rwrd*. Ad esempio, l'eliminazione del gruppo finale “-ia” e della vocale finale “-e” rende eguali “camic-ia” e “camic-e”, in quanto entrambe divengono “camic”; dal momento che queste due parole individuano categorie di attività economica diversamente codificate (la produzione di camicie corrisponde al codice 18.23.0; la produzione di camici corrisponde al codice 18.22.0) si è reso necessario rappresentare la differenza di significato in modo da raggiungere una corretta codifica. Dal momento che la produzione di camici rientra nella classe più ampia della produzione di abbigliamento da lavoro, la differenza di significato è stata ottenuta sostituendo “camice/i” con la coppia di parole “abbigliamento lavoro” nel file dati *Rwrd*.

Esempio: CAMICE *(diventa)* ABBIGLIAMENTO LAVORO
CAMICI *(diventa)* ABBIGLIAMENTO LAVORO

Quando nessuna delle parole appartiene a un particolare aggregato o non sia opportuno procedere a delle sostituzioni, si può procedere in altro modo e mantenere la desinenza di una delle parole, trascrivendola nel file dati *Excp*.

[13] Il gruppo vocalico “ia” non è sempre interamente desinenza: la “i” può essere tematica o radicale. (ad esempio: la “i” appartiene alla desinenza nella parola “camic-ia, camic-ie”, mentre è tematica, manca al plurale, nella parola “bilanc-i-a, bilanc-e”). Dal momento, che nel file dati *Sufx* non è possibile rappresentare le regole fonetiche, il gruppo “ia” è assunto sempre come desinenza.

4.2 Standardizzazione fonologica grammaticale[14]

Actr consente di eliminare, considerandoli "delimitatori", i segni di interpunzione e i segni ortografici compresi anche il simbolo di "$", il segno di "/ " e il segno di "*". Infatti, con Actr occorre definire un elenco di caratteri "validi" necessari alla composizione delle singole parole. I caratteri non riportati in questa lista sono considerati "delimitatori" (cioè caratteri non-validi) e come tali sono eliminati. Tuttavia, nel caso lo si ritenga necessario, Actr permette anche un intervento specifico di standardizzazione. Nell'allestimento del dizionario processabile Ateco, per alcuni tipi di parole, è stato necessario sottoporre a specifiche funzioni l'apostrofo e i punti compresi tra le lettere delle sigle.

4.2.1 Standardizzazione dei segni ortografici

- Eliminazione delle parole apostrofate in *Dstr*

L'apostrofo è eliminato, trascrivendo in uno specifico file dati, le parole che presentano questa forma di scrittura. In pratica gli articoli e le preposizioni articolate apostrofate sono escluse nel file dati relativo al processo definito *"Deletion String"(Dstr)* eseguito nella *Phrase Processing* (confrontare paragrafo 2.3.1.2, "Fase1 Step2:Elaborazione della frase").

Esempio: ALL' *(diventa) blank.*

Come si vede anche dall'esempio, il file dati *Dstr* consiste di una sola colonna (colonna 1-50). Infatti, è sufficiente trascrivere le parole (stringhe), con questo formato, perché siano cancellate, senza bisogno di altre indicazioni (confrontare paragrafo 2.3.4).
Questo tipo di trasformazione si rende praticamente necessario per ovviare alla possibilità che alcuni rispondenti possano omettere la parola apostrofata, mentre altri possano scriverla senza apostrofo e altri ancora possano scriverla correttamente con l'apostrofo.
Una frase del tipo "servizi all'agricoltura", può anche venire scritta sbrigativamente, come "servizi agricoltura" o più semplicemente può contenere un errore ortografico, come l'omissione dell'apostrofo, "servizi all agricoltura". La rimozione di "all'" apostrofata e di "all" senza apostrofo permette di giungere a un'unica descrizione standard capace di *matching* con tutte le altre. In casi come questo, per ottenere una standardizzazione, è necessario eliminare queste parole dal testo, sostituendo con un *blank* sia la forma apostrofata in *Dstr*, sia la forma senza apostrofo in *Rwrd.*

- Eliminazione dei punti nelle sigle in *Rstr*

Per quanto riguarda le sigle Actr ne permette la trasformazione in una apposito file dati *Replacement Strings (Rstr),* appartenente alla fase 1, step 2 della *Phrase Processing*. Le sigle, essendo costituite ciascuna da una stringa di lettere intervallate da punti, devono essere trasformate prima del *matching*. Infatti la eliminazione generalizzata dei punti, renderebbe ciascuna lettera indipendente dalle altre. Nel file dati *Rstr,* le sigle sono equiparate a parole costituite dalle lettere

---

La punteggiatura consiste dei segni di interpunzione e dei segni ortografici. I segni d'interpunzione sono: la virgola, il punto e virgola, i due punti, il punto fermo, il punto interrogativo e il punto esclamativo. I principali segni ortografici sono: le parentesi, le virgolette, il trattino, la lineetta, i puntini di sospensione e l'asterisco.

scritte in successione senza punti, né spazi. Il file dati *Rstr* è suddiviso in due colonne. Nella prima (colonna 1-50) è trascritta la stringa da sostituire, nella seconda (colonna 51-100), è trascritta la stringa che la sostituisce (confrontare paragrafo 2.3.4).

Esempio:

F.F.S.S. *(diventa)* FFSS

Anche questa semplice trasformazione non è meccanica e richiede sempre una valutazione contestuale: se sia più opportuno uguagliare la sigla alla sua identica forma senza punti, o piuttosto al corrispondente nome esteso.

Esempio

F.F.S.S *(diventa)* FERROVIESTATO

Comunque, è sempre necessario prevedere eventuali altre sostituzioni nei file dati dei processi successivi *Hyphenated Words (Hwrd)*, *Replacemet Words (Rwrd)*, *Double Words (Dwrd)*, in maniera coerente con quella adottata in *Rstr*.

Esempio:

Se, nella Rstr è stata stabilita la sostituzione

F.F.S.S. *(diventa)* FERROVIESTATO

allora nella Hwrd la sostituzione eventuale sarà:

FERROVIE-STATO *(diventa)* FERROVIESTATO

allora, nella Rwrd la sostituzione eventuale sarà:

FFSS *(diventa)* FERROVIESTATO

allora nella Dwrd la sostituzione eventuale sarà:

FERROVIE STATO *(diventa)* FERROVIESTATO

Per quanto riguarda, sia *Deletion Clauses,* sia *Replacement Strings,* sia gli altri processi della *Phrase Processing,* non occorre aggiungere altro a quanto già detto nella prima parte. Perciò si rimanda, direttamente, al paragrafo 2.3.1 e al paragrafo 2.3.2, dove sono descritti in modo completo.

- Eliminazione del trattino nelle parole composte in *Hwrd*

La soluzione adottata per l'applicazione Ateco non utilizza il processo *Hwrd*.
L'uso del file *Hwrd* si rende necessario **soltanto** quando il trattino sia stato incluso tra i caratteri "validi" nell'apposito file *Wchr*. Qualora ciò non sia stato fatto, per mantenere l'unione significativa di due parole è sufficiente avvalersi del processo *Dwrd*. **Si ricorda** che, per l'applicazione sull'Ateco, è stata adottata quest'ultima soluzione.
Esempio:

in questo caso la sostituzione avverrà nella Dwrd:
CAPO TRENO *(diventa)* CAPOTRENO

Per completezza, viene descritta anche la possibile utilizzazione del processo "Hyphenated Words". Quando il trattino compreso tra due parole (unione di due termini che esprimono un unico concetto) è un carattere "valido", può essere eliminato in uno specifico file dati del processo definito *"Hyphenated Words" (Hwrd)*, a livello dello step di *Word Processing* (confrontare il paragrafo 2.3.2.1, "Fase2. Step1: Elaborazione delle parole").
Il file dati *Hwrd* è suddiviso in quattro colonne (confrontare paragrafo 2.3.4). Nella prima colonna (colonna 1-60) è trascritta la parola composta completa di trattino e senza spazi (parola, trattino, parola). Nella seconda colonna (colonna 61-90) è trascritta in genere la medesima parola composta, senza trattino e senza spazi. I due termini della parola composta sono trascritti attaccati. Quando la parola composta è sostituita da un unico termine, questo è comunque trascritto nella seconda colonna. Quando la parola composta è sostituita da due termini, uno è scritto in seconda colonna (colonna 61-90) e l'altro in terza colonna (colonna 91-120). Nel caso, la parola composta sia sostituita da tre termini, oltre a compilare la seconda e terza colonna, si compila anche la quarta (colonna 121-150).

Esempio: CAPO-TRENO *(diventa)* CAPOTRENO.

Questo tipo di trasformazioni si rende necessario per mantenere l'unione significativa di due parole che esprimono un preciso significato solo in associazione l'una con l'altra, mentre, separatamente, ne esprimono un altro. Nella parola "capo-treno", se la rimozione automatica del trattino avesse luogo, si otterrebbero due parole "capo" e "treno" svincolate tra di loro. Il *match* potrebbe verificarsi con molte descrizioni che contengono una qualunque combinazione di queste parole e porterebbe a una codifica errata.

EMO-BANCHE *(diventa)* EMOBANCHE.

È altrettanto necessario prevedere anche la forma nella quale le parole siano scritte dal rispondente separate senza trattino e riportarle nel file dati Dwrd per sostituirle con la corrispondente parola unica.

Esempio: *il nome composto "emo banche" ottenuto con due parole separate scritte senza trattino è trascritto in* Dwrd

EMO BANCHE *(diventa)* EMOBANCHE.

Nel caso di un nome composto sinonimo di un altro nome composto si devono stabilire delle sostituzioni coerenti con la prima (confrontare paragrafo 4.3).

Esempio: *il nome composto "banche-sangue" sinonimo di "emobanche" è trascritto in Hwrd*:

BANCHE-SANGUE *(diventa)* EMOBANCHE.

Esempio: *il nome composto "banche sangue" sinonimo di "emobanche" è trascritto in Dwrd*:

BANCHE SANGUE *(diventa)* EMOBANCHE.

Esempio: *il nome composto "banchesangue" sinonimo di "emobanche" è trascritto in Rwrd*:

BANCHESANGUE *(diventa)* EMOBANCHE.

4.3 Standardizzazione dei significati

Dal momento che per Actr le parole sono semplicemente una sequenza di caratteri, i significati non esistono, se non come attribuzione di senso implicita. La certezza del significato riposa sull'assunto che esista una corrispondenza univoca tra oggetto grafico e parola. In questo modo, il significato viene a dipendere esclusivamente dalla sua forma di scrittura. Del resto, è anche evidente che, ai fini di un *matching* corretto, non si può prescindere dalla certezza che l'attribuzione dei significati alle parole (vale a dire, alla loro forma di scrittura) avvenga sulla base di una standardizzazione linguistica controllata. Per questo motivo, le differenze di scrittura delle parole devono non solo essere ricondotte ad una sola forma grafica invariante, ma anche ad una parola che assuma un medesimo significato, comunque valido nei vari punti del dizionario.
Nel testo considerato, si sono individuate delle differenze di scrittura che possono essere ricondotte a tre tipi di parole: sinonimi (anche solo contestuali), abbreviazioni (spesso arbitrarie), nomi di oggetti o parti di essi in luogo della categoria di appartenenza.

- Parole effettivamente diverse che rimandano ad uno stesso significato: designano uno stesso oggetto, attività, concetto in modo da poter essere considerate, anche solo contestualmente, dei sinonimi. Ad esempio, possono considerarsi sinonimi contestuali, le parole "fabbricazione" e "produzione".
- Parole che, o perché abbreviate (in modo frequentemente arbitrario) o perché scritte con particolari segni grafici, si discostano dalla forma prevalente di scrittura. Nella realtà del testo esaminato le parole presentano spesso entrambe le caratteristiche. Ad esempio, l'abbreviazione della parola "produzione" in "prod" può essere considerata arbitraria, sia da un punto di vista grammaticale, sia da un punto di vista dell'ambiguità di significato contestuale. Per quanto riguarda le sigle, si rimanda a quanto già detto sia nel paragrafo dal 2.3.1.2., nella prima parte, sia nel paragrafo 4.1. Infatti, non rientrano in questa casistica, perché non si tratta di compiere un'attribuzione di significato, ma piuttosto di stabilire una corrispondenza tra la forma di scrittura estesa e quella contratta di un significato ufficialmente condiviso.
- Parole che designano oggetti particolari altrimenti individuati dal nome della classe di appartenenza. In questo caso, un oggetto o una sua parte designa il tutto. Ad esempio, la classe di oggetti definita dalla parola "infissi", può essere indicata attraverso il nome di un elemento di appartenenza, come "porte", "finestre", o alcune parti di questi, come "telai", "stipiti".

Esempio: b)
Fabbricazione (di) infissi (in) legno
Produzione (di) porte e finestre (di) legno

Esempio: bb)
Fabbricaz. di infissi in legno
Produz. di porte e finestre di legno

Come si vede nell'esempio b) la medesima attività e i medesimi manufatti sono descritti con parole diverse: la prima attraverso i sinonimi “produzione” e “fabbricazione” , i secondi, ora attraverso il nome della classe “infissi”, ora attraverso il nome degli oggetti inclusi nella classe “porte e finestre”.
Inoltre, nell'esempio bb) si può osservare come le parole che descrivono l'attività, anche se, in questa frase, correttamente abbreviate, tuttavia rappresentino una variazione rispetto alla loro forma estesa.
Il modo che Actr permette di usare per risolvere questi casi di variabilità consiste nell'assegnare uno stesso grafema ai diversi tipi di scrittura. Nel caso dell'esempio, questo significa stabilire un'identità tra le parole “fabbricazione”, “fabbricaz.” “produz” e la parola “produzione” in modo da sostituire i tre diversi grafemi con uno solo.

4.3.1 Standardizzazione dei sinonimi e costruzione di classi di oggetti

Ciò avviene principalmente attraverso i due processi della *Word Processing*, uno definito *Replacement Words*, l'altro definito *Double Words*. In base a semplici regole di assegnazione, risulta possibile sostituire parole e indirettamente attribuire e modificare significati.
Ad esempio le descrizioni degli esempi b) e bb) sono trasformate in modo da essere lette come se fossero identiche:

se, nella Rwrd vengono stabilite le sostituzioni:

FABBRICAZIONE	*(diventa)*	PRODUZIONE
FABBRICAZ	*(diventa)*	PRODUZIONE
PRODUZ	*(diventa)*	PRODUZIONE
FINESTRA	*(diventa)*	INFISSO
FINESTRE	*(diventa)*	INFISSO
INFISSI	*(diventa)*	INFISSO
PORTA	*(diventa)*	INFISSO
PORTE	*(diventa)*	INFISSO;

allora, data la standardizzazione grammaticale, si avrà una sola descrizione:

“Produzion(e) infiss(o) legn(o)”.

Questo insieme di parole, per le assegnazioni attribuite ai singoli termini, risulta uguale a qualunque altro insieme ottenuto dalla combinazione di quegli stessi termini. Questo significa che, dati quei termini, la descrizione così ottenuta (ordinata nel *sort:* Infiss Legn Produzion) può “leggere” tutte le altre.

4.3.2 Standardizzazione delle abbreviazioni

Situazioni più complicate sono rappresentate da descrizioni nelle quali le parole non sono abbreviate correttamente: spesso, la parola ridotta al di sotto del tema o addirittura della radice, rende impossibile una univoca attribuzione di significato.

Esempio: Fab. infissi (in) legno.

Prod. porte e finestre (di) legno.

Come si vede, le parole “fab.” e “prod.” non possono essere trasformate con certezza in “produzione”. Infatti, potrebbero essere abbreviazioni anche di altre parole, come ad esempio “fabbricato” e “prodotto”. In questi casi, occorre fare una valutazione dell'uso contestuale. Occorre verificare se, nel contesto considerato, questo tipo di abbreviazioni effettivamente assume significati diversi che non permettono una trasformazione univoca, oppure se assume significato in modo uniforme.
Nel caso si osservi che ad una abbreviazione eccessivamente contratta corrisponde un solo significato, sulla base di questa evidenza empirica, si può procedere all'assegnazione di quel solo significato, non considerando gli altri linguisticamente possibili perché non utilizzati.
Rientra in questa casistica la parola “fab.” dell'esempio precedente. Nel contesto analizzato, “fab.” assume con tale frequenza il significato di “fabbricazione” da indurre a sceglierlo come significato univoco.
Per la sostituzione stabilita precedentemente di “fabbricazione” con “produzione”, la parola “fab.” è direttamente assegnata a “produzione”.

Esempio:

se si assume l’identità di significato: Fab = Fabbricazione,

e se nella Rwrd si stabilisce la sostituzione

FABBRICAZIONE *(diventa)* PRODUZIONE;

allora sarà anche valida la sostituzione:

FAB *(diventa)* PRODUZIONE.

Nei casi nei quali le parole abbreviate risultano incomprensibili, o non ascrivibili ad un significato univoco, si può ricorrere ad un'altra soluzione.
Come si sarà osservato, la possibilità di attribuire un significato alle parole arbitrariamente abbreviate è maggiore quando non siano considerate singolarmente, ma abbinate alla parola che le segue o che le precede.
La parola “prod.” dell'esempio precedente, a differenza della parola “fab.”, assume, con uguale frequenza nel testo, i due diversi significati di “produzione” e di “prodotto”.
Si rende così necessario, considerare la parola “prod” insieme alla parola seguente; nel caso dell'esempio, “porte”.
La coppia “prod. porte” ha un significato più stabile, che permette di scegliere l'assegnazione alla parola “produzione”, escludendo l'altra.

Esempio:

se, nella Rwrd viene indicata la sostituzione:

PORTA *(diventa)* INFISSO

PORTE *(diventa)* INFISSO.

Allora, nella Dwrd la sostituzione sarà:

PROD INFISSO *(diventa)* PRODUZIONE INFISSO.

Tuttavia, bisogna osservare che, in questo modo, l'assegnazione è valida solo per le parole specificate nella coppia, nella forma e nell'ordine nei quali gli abbinamenti sono trascritti.

L’assegnazione ha valore più ristretto e in genere vincolato ad una attività o prodotto particolari. Invece, le sostituzioni che in prima colonna hanno una sola parola (*Rwrd*) sono svincolate dalla posizione e sempre valide in qualunque punto del testo.
Anche per questa parte, si rimanda all’esposizione che, di questi stessi argomenti, viene fatta nella “Guida al Parsing”, nel paragrafo 2.2 e seguenti.

4.4 Effetti dell’ordine di esecuzione dei processi di standardizzazione: definizione della regola di trascrizione

Le modificazioni di tipo grammaticale, sia l’esclusione degli articoli, delle preposizioni, delle congiunzioni e degli avverbi (paragrafo 4.1.1), sia la standardizzazione delle parole del genere e del numero (paragrafo 4.1.2), avvengono in due momenti e in due processi distinti dello step di *Word Processing.* Le prime, definite nel processo *Replacement Words*, sono eseguite all'inizio; le seconde, definite nel processo *Suffixes*, sono eseguite alla fine, immediatamente prima del *Post-Processing* (confrontare nella prima parte, il paragrafo 2.2 e seguenti.). Questa posizione dei processi, iniziale per *Replacement Words*, finale per *Suffixes*, crea una condizione vincolante per il metodo di trascrizione delle parole nei *data file di parsing*. Infatti, occorre tenere conto che i processi della *Word Processing* sono eseguiti su un elenco di parole ancora non depurate del genere e del numero. Da questo segue che, per generalizzare le sostituzioni, sia necessario trascrivere ogni parola da sostituire, in ciascuna forma del genere e del numero (maschile singolare, maschile plurale; femminile singolare, femminile plurale). Tuttavia, si è valutato che una diversa distribuzione dell’ordine dei processi comporta un’elevata possibilità di errore per chi aggiorna l’applicazione di codifica automatica. Infatti, nella gestione del processo *Rwrd* ci si troverebbe costretti a ripercorrere mentalmente la stessa logica che Actr segue in *Sufx* e *Prfx.*
Questa condizione costituisce **“la regola di trascrizione”** di tutte le parole in qualunque processo di sostituzione della *Word Processing*.
Per chiarire meglio la necessità di quella che abbiamo definito “regola di trascrizione”, occorre ricordare che le sostituzioni sono valide esclusivamente nel modo in cui le parole sono trascritte. Sono assegnazioni di grafemi a grafemi, che non supportano alcun significato, se non quello attribuitogli dall'esterno. In questo senso, il genere e il numero di ciascuna parola costituiscono grafemi non coincidenti. Per generalizzare le sostituzioni è quindi necessario ricondurli tutti ad una forma unica invariante. A questo scopo, vengono definiti dei criteri applicativi specifici per la compilazione dei file dati *Rwrd* e *Dwrd*.

4.5 Applicazione della regola di trascrizione nel file dati Replacement Words

La funzione *Replacement Words* (*Rwrd)* permette di sostituire una sola parola, alternativamente, con una sola parola, con due parole, con nessuna parola. Perciò la sostituzione è valida in una sola direzione da sinistra a destra, e non viceversa. Il caso di sostituzione con nessuna parola equivale a una cancellazione.

Esempio:

INDUMENTO *(diventa)* ABBIGLIAMENTO *(non viceversa)*

Il file dati *Rwrd* si presenta suddiviso in tre colonne. Nella prima (colonna 1-30), sono trascritte le parole da sostituire. Nelle altre due sono trascritte le parole che la sostituiscono. Quando è assegnata una sola parola, questa è riportata nella seconda colonna (colonna 31-60). Quando è assegnato un *blank*, non vanno compilate né la seconda (colonna 31-60), né la terza (colonna 61-90). Quando è assegnata una coppia di parole, queste sono trascritte rispettivamente nella seconda (colonna 31-60) e nella terza (colonna 61-90) colonna (confrontare paragrafo 2.3.4).
Per generalizzare l'assegnazione, **le parole che figurano in prima colonna vanno trascritte in tutte le forme del genere e del numero**. Per quel che riguarda il secondo termine, i sostantivi e gli aggettivi che hanno un'unica forma per entrambi i generi vanno sempre trascritti al singolare, mentre gli aggettivi declinabili vanno trascritti al singolare maschile. La parola posta in seconda colonna è essa stessa sottoposta a particolari regole di trascrizione del suo genere e del suo numero, come verrà spiegato nel paragrafo 4.5.2, alla lettura del quale si rimanda.

- Sostituzione con un sostantivo:

Esempio:

INDUMENTO *(diventa)* ABBIGLIAMENTO
INDUMENTI *(diventa)* ABBIGLIAMENTO
VESTIARIO *(diventa)* ABBIGLIAMENTO
VESTIARI *(diventa)* ABBIGLIAMENTO
VESTITO *(diventa)* ABBIGLIAMENTO
VESTITI *(diventa)* ABBIGLIAMENTO.

- Sostituzione con una coppia di sostantivi:

Esempio:

CAMICE *(diventa)* ABBIGLIAMENTO LAVORO
CAMICI *(diventa)* ABBIGLIAMENTO LAVORO
DIVISA *(diventa)* ABBIGLIAMENTO LAVORO
DIVISE *(diventa)* ABBIGLIAMENTO LAVORO.
UNIFORME *(diventa)* ABBIGLIAMENTO LAVORO
UNIFORMI *(diventa)* ABBIGLIAMENTO LAVORO

- Sostituzione con un aggettivo che ha una forma unica per entrambi i generi:

Esempio:

ARTIGIANA *(diventa)* ARTIGIANALE
ARTIGIANO *(diventa)* ARTIGIANALE
ARTIGIANE *(diventa)* ARTIGIANALE
ARTIGIANI *(diventa)* ARTIGIANALE.

- Sostituzione con un sostantivo e un aggettivo che ha una forma unica per entrambi i generi:

Esempio:

date le due sostituzioni del plurale con il singolare:
ALLEVAMENTI*(diventa)* ALLEVAMENTO e VOLATILI *(diventa)* VOLATILE,

allora sarà:
AVICOLTURA *(diventa)* ALLEVAMENTO VOLATILE
AVICOLTURE *(diventa)* ALLEVAMENTO VOLATILE.

- Sostituzione con un aggettivo completamente declinabile:

Esempio:
AGRARIA *(diventa)* AGRICOLO
AGRARIO *(diventa)* AGRICOLO
AGRARIE *(diventa)* AGRICOLO
AGRARI *(diventa)* AGRICOLO.

4.5.1 Accorgimenti nella trascrizione nel file dati *Rwrd*

Non è forse inutile, ancora una volta, osservare che, in Actr, le sostituzioni sono valide così come vengono trascritte, riga per riga, indipendentemente le une dalle altre. Sarebbe sbagliato attendersi un controllo semantico da un software che si fonda su un criterio di riconoscimento grafico delle parole.
Per questo motivo, una particolare attenzione è stata rivolta alla trascrizione delle sostituzioni per evitare sia gli errori ortografici, sia gli errori logici.

- Errori ortografici:

Esempio:
AGRARA *(diventa)* AGRICOLO
AGRARIA *(diventa)* AGICOLO.

- Errori logici:

Esempio:
AGRARIO *(diventa)* AGRICOLO
AGRICOLO *(diventa)* AGRARIO

ARTIGIANO *(diventa)* ARTIGIANALE
ARTIGIANO *(diventa)* OPERAIO.

Le sostituzioni errate, anche se incongruenti con quelle della riga immediatamente seguente o precedente, sono eseguite comunque, insieme a tutte le altre creando importanti disfunzioni nel *matching*. Ad esempio, la sostituzione di “agraria” con “agicolo”, rende impossibile qualunque riconoscimento della parola “agicolo” non più riconducibile a “agricolo”. Meno grave, è il caso in cui “agrara” è assegnato a “agraria”, perché, la validità della classe di attribuzione sussiste comunque.
Dal momento che tutte le sostituzioni sono eseguite nell'ordine con il quale vengono trascritte nel file dati, può anche accadere che alcune non siano eseguite se la trasformazione è incongruente con

l'ordine. Ad esempio, la sostituzione di “artigiano” con “artigianale”, eseguita in ordine alfabetico, avviene prima di quella errata che assegna “artigiano” a “operaio”. Al momento di questa seconda trasformazione, il termine “artigiano” non esiste più e dunque la sostituzione non è eseguita.
Non deve trarre in inganno il fatto che la sostituzione non eseguita sia quella errata: sarebbe successa la stessa cosa anche se fosse stata corretta. Ciò che decide è l'ordine di esecuzione.
Riassumendo brevemente, si può dire che gli errori di trascrizione ortografici sono meno gravi quando interessano il primo termine e più gravi quando interessano il secondo; gli errori logici dipendono dalla posizione nell'ordinamento alfabetico e sono sempre fortemente distorcenti sia che interessino il primo che il secondo termine.
Inoltre, per ottenere una gestione del file il più possibile ordinata, è sembrato opportuno ordinare le sostituzioni ponendo il secondo termine in ordine alfabetico. (Questo ordinamento dei file di parsing deve essere ripetuto ogni volta che si provvede ad una integrazione e/o modifica. Inoltre, si deve tenere conto che l'elenco ordinato delle sostituzioni non ha solo il vantaggio di permettere una consultazione rapida durante le operazioni di integrazione del file dati, ma soprattutto ha un valore funzionale, in quanto rappresenta l'ordine di esecuzione delle sostituzioni).

4.5.2 Regole di trascrizione del genere e del numero nel file dati *Rwrd*

Inoltre, come si sarà osservato negli esempi precedenti, per le parole assegnate (poste in seconda colonna) non sono indicate le sostituzioni con le proprie forme del genere e del numero. Queste non sarebbero necessarie perché la successiva perdita della vocale finale (processo *Suffixes Words*) uniforma comunque tutti i casi (ad esempio, “agricola, agricole, agricoli” e “agricolo” diventano tutte “agricol”). In realtà, si può usufruire correttamente di questa condizione solo per le parole del file dati *Rwrd* che non compaiono in quello *Dwrd*.
Dal momento che i file dati *Rwrd* e *Dwrd* possono, nel tempo, andare incontro a continue integrazioni e parziali modificazioni, non si tiene conto di questa differenza; e così, **anche tutte le parole poste in seconda e terza colonna in *Rwrd* sono sempre uniformate al proprio singolare** per evitare possibili confusioni in eventuali trascrizioni successive. Infatti, il riportare tutto al maschile singolare comporta un alleggerimento delle trasformazioni da gestire successivamente nel processo *Dwrd*.

Esempio:

AGRARIA *(diventa)* AGRICOLO
AGRARIO *(diventa)* AGRICOLO
AGRARIE *(diventa)* AGRICOLO
AGRARI *(diventa)* AGRICOLO
AGRICOLA *(diventa)* AGRICOLO
AGRICOLE *(diventa)* AGRICOLO
AGRICOLI *(diventa)* AGRICOLO

4.6 Effetti della posizione sequenziale dei processi Replacement Words e Double Words nell'applicazione della regola di trascrizione

Per meglio chiarire quanto appena detto riguardo alla necessità, o meno, di indicare nel file dati *Rwrd* le sostituzioni per il genere e il numero delle parole in seconda e terza colonna, occorre fare delle considerazioni. In primo luogo, occorre dire che i due processi *Rwrd* e *Dwrd* sono sequenziali: il processo *Double Words* effettua le proprie sostituzioni dopo quelle già eseguite in *Replacement Words*. Questo significa che *Double Words* non "legge" più, ad esempio, la parola "indumento", ma direttamente la parola assegnata "abbigliamento".

Esempio: ABBIGLIAMENTO *ha già sostituito*
ABITO/I; INDUMENTO/I; VESTIARIO/I; VESTITO/I.

In secondo luogo, occorre precisare che i due file dati sono indipendenti per quanto riguarda la compilazione, mentre logicamente sono interdipendenti. Le regole di trascrizione valide per la compilazione di *Rwrd* lo sono anche per la compilazione di *Dwrd*: in ogni caso, le sostituzioni sono valide esclusivamente, così come risultano trascritte, riga per riga, indipendentemente le une dalle altre. Per quanto riguarda le relazioni tra i due file dati, occorre osservare che, da un punto di vista logico, *Rwrd* e *Dwrd* rappresentano un unico insieme nel quale le sostituzioni di una sola parola sono eseguite prima di quelle di due parole. Questo significa che tutti i controlli di congruenza logica e di corretta trascrizione, validi per un solo file dati, devono essere estesi ai due file dati considerati contemporaneamente come fossero un unico file. Ad esempio, la descrizione "indumenti da cerimonia" al momento della esecuzione del processo di *Double Words* ha già subito delle trasformazioni di tipo generale (perdita della preposizione "da") e di tipo particolare (sostituzione di "indumento" con "abbigliamento"). A questo punto è diventata "abbigliamento cerimonia". Queste parole vanno riportate in *Dwrd*, così come risultano scritte dopo le precedenti trasformazioni, perché altrimenti non verrebbero eseguite.

4.7 Applicazione della regola di trascrizione nel file dati Double Words

La funzione *Double Words (Dwrd)* permette di sostituire una coppia di parole, alternativamente, con una sola parola, con due parole, con nessuna parola. Perciò la sostituzione è valida in una sola direzione da sinistra a destra, e non viceversa. Il caso di sostituzione con nessuna parola equivale a una cancellazione.
Il file dati *Dwrd* si presenta suddiviso in quattro colonne. Nella prima (colonna 1-30) e nella seconda colonna (colonna 31-60), è trascritta la coppia di parole da sostituire. Nelle altre due sono trascritte le parole che la sostituiscono. Quando è assegnata una sola parola, questa è riportata nella terza colonna (colonna 61-90). Quando alla coppia è assegnato un *blank*, non vanno compilate né la terza (colonna 61-90), né la quarta (colonna 91-120). Quando è assegnata una coppia di parole, queste sono trascritte rispettivamente nella terza (colonna 61-90) e nella quarta (colonna 91-120) colonna (confrontare paragrafo 2.3.4).

- Sostituzione di una coppia di parole con un'altra coppia:

Esempio:

ABBIGLIAMENTO CERIMONIA *(diventa)* ABBIGLIAMENTO ADULTO;

COLLABORATORE FAMIGLIA *(diventa)* COLLABORATORE DOMESTICO

- Sostituzione di una coppia di parole con una sola parola:

Esempio:

BANCHE SANGUE *(diventa)* EMOBANCHE.

- Sostituzione di una coppia di parole con una "parola unica" creata appositamente:

Esempio:

NON DISTILLATO *(diventa)* NONDISTILLATO.

Queste trasformazioni si rendono necessarie non per una esigenza grammaticale astratta. Derivano dalle caratteristiche del contesto viste esclusivamente in funzione di un efficace *matching* ai fini di una corretta codifica automatica. Si ricorre al file dati *Dwrd* per motivi di opportunità: dal momento che, per evidenti ragioni di significato, non è possibile sostituire direttamente "cerimonia" con "adulto", né tantomeno "familiare" con "domestico".
Così ad esempio, l'espressione "abbigliamento cerimonia", attraverso la sostituzione con "abbigliamento adulto", viene ricondotto alla relativa categoria di inclusione che, nella classificazione Ateco, corrisponde ad una precisa categoria del commercio al dettaglio (*Commercio al dettaglio confezioni per adulti* codice 52.42.1).
In modo analogo, l'espressione "collaboratore familiare" è sostituta con "collaboratore domestico" perché è solo una variante di quest'ultima: non è semplicemente inclusa, è addirittura la stessa categoria e come tale deve essere codificata (*Collaboratore domestico* codice 95.00.0).
Invece, il motivo per cui l'espressione "banche (a) sangue" (codice 85.14.4) viene trasformata nella parola unica "emobanche", deriva dalla necessità di vincolare, in modo il più possibile non ambiguo, la parola "banca/he" al settore specifico di attività economico-finanziaria al fine di evitare un'errata codifica automatica, qualora la parola "banca" e la parola "sangue" venissero abbinate, indipendentemente l'una dall'altra, nel processo di *matching*.

4.7.1 Accorgimenti nella trascrizione nel file dati *Dwrd*

Per quanto riguarda il file dati *Dwrd* valgono le regole di trascrizione nel file dati *Rwrd*. Si rimanda perciò alla lettura del paragrafo 4.5.1 .Tuttavia sussistono alcune differenze. In particolare, una maggiore difficoltà nella compilazione del file dati *Dwrd* risiede nel fatto che le operazioni di verifica della correttezza ortografica e della congruenza logica devono essere effettuate su un numero di parole maggiore.
Infatti i controlli vanno eseguiti su ciascuna parola sia della coppia che figura in prima e seconda colonna, sia di quelle che figurano in terza e quarta colonna.

In primo luogo, occorre verificare, per ciascuna parola, se questa è stata, o meno, soggetta a sostituzione nel file dati *Rwrd*. Questa distinzione preliminare permette di eseguire le operazioni di controllo ordinatamente secondo uno schema preciso come indicato qui di seguito.
La parola è già stata sostituita con un'altra parola/e in Rwrd? Sì.

Allora, occorre prima verificare nel file dati *Rwrd*:

- qual è la parola (posta in seconda colonna) o la coppia di parole (poste in seconda e terza colonna) con la/e quale/i è stata sostituita;

- se la parola/e che sostituisce (poste in seconda e terza colonna) è stata trascritta al singolare;

- se la parola (posta in prima colonna) è stata trascritta e quindi sostituita in tutti i casi;

- se sono state trascritte le sostituzioni di tutti i casi anche della parola/e (posta in seconda e terza colonna) con la corrispondente forma singolare.

La parola è già stata sostituita con un'altra parola in Rwrd? No.

Allora, occorre verificare nel file dati *Rwrd*:

- se siano già state trascritte le sostituzioni per tutti i casi (posti in prima colonna) della parola con la sua forma singolare (posta in seconda colonna) che è quella da ritrascrivere nel file dati *Dwrd*: in prima o seconda colonna, quando è necessaria un'ulteriore sostituzione, in terza o quarta colonna, quando rappresenta una definizione valida a raggruppare una certa classe di oggetti.

Una volta effettuato questo primo tipo di controlli sulla coerenza delle trascrizioni, una particolare attenzione deve essere rivolta alle parole che definiscono delle classi di oggetti o di significati (è evidente che questo tipo di controllo richiede, oltre alla verifica della congruenza logica, anche un'attività di documentazione sul relativo settore di attività economica).
Per questo motivo, è opportuno verificare se la parola scelta per definire una classe nel file dati *Rwrd* sia stata identicamente trascritta in *Dwrd*, evitando, così, di indicare un'altra parola per raggruppare lo stesso tipo di oggetti già incluso nella classe creata nel file dati *Rwrd.* Altrimenti, oggetti di uno stesso tipo verrebbero a costituire insiemi diversi (non a caso, in terza colonna in *Dwrd* sono ammesse anche sostituzioni con singole parole, in modo da mantenere una coerenza con le assegnazioni di *Rwrd*.).
Prendiamo ad esempio le trasformazioni seguenti.

Date le possibili diverse denominazioni di uno stesso oggetto:
abito/i cerimonia,
indumento/i cerimonia,
vestiario/i cerimonia,
vestito/i cerimonia;

per definire la classe "abbigliamento", nel file dati *Rwrd* sono state effettuate le seguenti sostituzioni con la parola "abbigliamento":

ABBIGLIAMENTI *(diventa)* ABBIGLIAMENTO
ABITO *(diventa)* ABBIGLIAMENTO
ABITI *(diventa)* ABBIGLIAMENTO
INDUMENTO *(diventa)* ABBIGLIAMENTO
INDUMENTI *(diventa)* ABBIGLIAMENTO
VESTIARIO *(diventa)* ABBIGLIAMENTO
VESTIARI *(diventa)* ABBIGLIAMENTO
VESTITO *(diventa)* ABBIGLIAMENTO
VESTITI *(diventa)* ABBIGLIAMENTO.

- Nel file dati *Dwrd* viene effettuata un'attribuzione ad una classe già definita nel file dati *Rwrd*:

Esempio:
ABBIGLIAMENTO FIRMATO *(diventa)* ABBIGLIAMENTO.

- Nel file dati *Dwrd* viene effettuata una sostituzione ulteriore dopo la prima sostituzione nel file dati *Rwrd*:

Esempio:
ABBIGLIAMENTO CERIMONIA *(diventa)* ABBIGLIAMENTO ADULTO.

Il motivo che ha reso necessario ottenere due definizioni diverse, "abbigliamento" e "abbigliamento adulto", è dovuto al fatto che, nella classificazione "Ateco", queste due definizioni individuano oggetti appartenenti a categorie di attività differenti e perciò identificate da codici diversi. La descrizione "abbigliamento firmato" non è significativa per la classificazione dell'attività economica, in quanto non individua nessuna particolare classe di oggetti , a parte l'abbigliamento. La parola "firmato" non viene perciò mantenuta. La definizione "abbigliamento" risulta generica e quindi compare sia nelle descrizioni della produzione che del commercio all'ingrosso e al dettaglio (nella classificazione Ateco, rispettivamente, codice 18, codice 51.42, codice 52.42); invece, la seconda definizione "abbigliamento adulto" si riscontra nelle descrizioni di una precisa categoria del commercio al dettaglio (*Commercio al dettaglio confezioni per adulti* codice Ateco 52.42.1). Perciò, è indispensabile che rimangano distinte linguisticamente, in modo da permettere una corretta operazione di *matching* e di corretta codifica.
I controlli relativi alla correttezza della classe di attribuzione devono ovviamente essere eseguiti anche in senso inverso: da *Dwrd* a *Rwrd*.
Nel caso in cui, una certa classe di attribuzione sia stata definita nel file dati *Dwrd*, in ordine di tempo, prima che in *Rwrd*, occorre verificare:

- se la classe già definita nel file dati *Dwrd* sia stata identicamente trascritta in *Rwrd*, evitando di indicare un'altra parola (o coppia di parole) per raggruppare lo stesso tipo di oggetti, già incluso nella classe creata nel file dati *Dwrd.* Altrimenti, oggetti di uno stesso tipo verrebbero a costituire insiemi diversi (non a caso *Rwrd*, ammette anche sostituzioni con coppie di parole, in modo da poter mantenere una coerenza con le assegnazioni di *Dwrd*);
- che la parola-classe (o coppia di parole-classe) da ritrascrivere in *Rwrd* occupi esclusivamente la seconda colonna (o seconda e terza colonna), in quanto la sostituzione può rappresentare solamente un'implementazione della classe medesima.

Prendiamo ad esempio le trasformazioni seguenti.

Date le diverse denominazioni di uno stesso oggetto:

indumento cameriere,
vestiario cameriere,
vestito cameriere,
abito cameriere.

Se, nella Rwrd vengono indicate le sostituzioni:

ABBIGLIAMENTI	*(diventa)* ABBIGLIAMENTO
ABITO	*(diventa)* ABBIGLIAMENTO
ABITI	*(diventa)* ABBIGLIAMENTO
INDUMENTO	*(diventa)* ABBIGLIAMENTO
INDUMENTI	*(diventa)* ABBIGLIAMENTO
VESTIARIO	*(diventa)* ABBIGLIAMENTO
VESTIARI	*(diventa)* ABBIGLIAMENTO
VESTITO	*(diventa)* ABBIGLIAMENTO
VESTITI	*(diventa)* ABBIGLIAMENTO

Se, nella Dwrd viene definita la classe:

ABBIGLIAMENTO CAMERIERE *(diventa)* ABBIGLIAMENTO LAVORO,

allora anche nella Rwrd la classe dovrà essere la stessa:

CAMICE *(diventa)* ABBIGLIAMENTO LAVORO.
CAMICI *(diventa)* ABBIGLIAMENTO LAVORO.

Anche in questo caso, il motivo che ha reso necessario ottenere la definizione "abbigliamento lavoro" è dovuto al fatto che nella classificazione Ateco la confezione di indumenti da lavoro corrisponde ad un preciso codice (codice 18.21). È stato necessario costruire una classe (abbigliamento lavoro) in modo che i nomi dei singoli indumenti con i quali viene descritta l'attività di confezionamento corrispondano tutti alla medesima definizione e parimenti al medesimo codice.
A conclusione di quanto detto, non è forse inutile aggiungere che entrambi i controlli (sia nel caso in cui una classe sia stata definita in *Rwrd*, sia nel caso in cui una classe sia stata definita in *Dwrd*) vanno eseguiti in maniera incrociata, contemporaneamente, perché, solo così è possibile evitare errate duplicazioni di una medesima classe.

4.7.2 Regole di trascrizione del genere e del numero nel file dati *Dwrd*

Nel file dati *Dwrd*, come in quello *Rwrd*, tutte le parole devono essere trascritte al singolare (ad eccezione di quelle che nel file dati *Excp* sono sostituite a favore del plurale, come già detto nel paragrafo 4.1.3). Per questo motivo la concordanza tra sostantivo e aggettivo non è sempre rispettata. Infatti, per coerenza con la regola di trascrizione, l'aggettivo declinabile deve essere trascritto al singolare maschile, anche quando il sostantivo cui si riferisce è femminile. Questo effetto, linguisticamente errato, è una conseguenza della strategia di standardizzazione; tuttavia, non

crea alcun problema nel processo finale di *matching*, perché, questo avviene, quando le parole sono ormai private della vocale finale.

Esempio:

nel file dati Dwrd sono trascritte le coppie di parole:

Nome	*Aggettivo*
APPARECCHIATUR**A** *(s.f.)*	ELETTRIC**O** *(s..m.)*
MATER**IA** *(s.f.)*	PLASTIC**O** *(s.m.).*

dopo l'esecuzione della funzione Sufx diventano:

APPARECCHIATUR	*ELETTRIC*
MATER	PLASTIC

Dall'esempio, si può vedere come risultano modificate le parole "apparecchiatura elettrico" e "materia plastico" dopo aver perso la vocale finale, prima di essere sottoposte al *match*.

5 Il sistema Blaise

5.1 Introduzione al sistema Blaise

Blaise, sviluppato e commercializzato da Statistics Netherlands, è un sistema per l'acquisizione dati assistita da computer (CAI – Computer Assisted Interviewing), ampiamente utilizzato non soltanto in Istat, ma in numerosi Istituti nazionali di statistica. È composto da una serie di moduli che supportano molte funzioni, tutte inerenti l'acquisizione dati, quali, per esempio:

- la messa a punto di questionari elettronici;
- la trasportabilità degli stessi in funzione di diverse tecniche di rilevazione;
- la gestione dei contatti per le rilevazioni telefoniche;
- l'interfaccia con altri software per l'analisi dei dati;
- la possibilità di accedere a dati in ambienti applicativi esterni a Blaise.

Il modulo di codifica assistita rientra nell'ambiente applicativo di sviluppo dei questionari elettronici.

Il questionario elettronico, chiamato in termini tecnici ***DATAMODEL***, è composto da due sezioni principali:

- ***FIELDS*** → in questa sezione sono elencate tutte le variabili oggetto di rilevazione e sono specificate le caratteristiche e la tipologia di ciascuna variabile. Viene definito in pratica se trattasi di variabili numeriche, alfabetiche o precodificate, viene esplicitato il dominio di ciascuna variabile e specificato se è ammessa o meno la mancata risposta, il non so ed il rifiuto.
- ***RULES*** → in questa sezione vengono elencate le variabili definite nella sezione precedente nell'ordine in cui si intende proporre i quesiti nel corso dell'intervista; vengono inoltre esplicitati tutti i controlli inerenti i range delle variabili e quelli di coerenza tra quanto risposto ai diversi quesiti previsti nel questionario.

Potranno essere oggetto di codifica assistita le variali definite nella sezione *FIELDS* come alfabetiche; il modulo di codifica assistita sarà richiamato nell'ambito della sezione *RULES*.

Come già specificato, la codifica assistita può essere effettuata nel corso dell'intervista, oppure in una fase successiva. Nel primo caso, non appena viene posto il quesito inerente la variabile testuale da codificare e viene digitata la risposta nell'apposito campo, si richiama il modulo di codifica assistita per l'individuazione del codice da assegnare.
Nel caso, invece, in cui si codifichi in un momento successivo rispetto alla rilevazione, possono verificarsi due situazioni:

- i dati sono stati acquisiti avvalendosi del Blaise, quindi, a prescindere dalla tecnica di rilevazione utilizzata, esisterà un *DATAMODEL* corrispondente al questionario di rilevazione, nell'ambito del quale dovrà essere attivata la funzione di codifica assistita in corrispondenza della variabile testuale da codificare;

- i dati sono stati acquisiti senza avvalersi del sistema Blaise; sarà quindi necessario caricare in ambiente Blaise i dati da codificare. È possibile sviluppare un *DATAMODEL* corrispondente a tutto il questionario di rilevazione, in cui saranno caricati i dati completi delle interviste, oppure ci si può limitare ad implementare un *DATAMODEL* che contenga, oltre ovviamente all'identificativo dell'intervistato, le variabili testuali da codificare ed eventuali altre variabili a queste correlate (le cui risposte possono essere di supporto al codificatore per la selezione del codice corretto).

Come descritto di seguito, sia il caricamento di un file ASCII in ambiente Blaise che lo scarico in ASCII di un file Blaise sono realizzabili tramite altrettanti comandi nell'ambito di un apposito modulo chiamato MANIPULA.

Relativamente, infine, al processo in base a cui il codificatore può pervenire all'individuazione del codice da assegnare, Blaise consente la navigazione nel dizionario secondo **tre modalità**:
- **secondo l'albero** → il codificatore visualizza in un primo tempo la classificazione a livello dei rami gerarchicamente più elevati (ad esempio, se ci si riferisse alla classificazione dei Comuni, potremmo avere al primo livello le Regioni, al secondo le Province ed infine i Comuni), quindi seleziona il ramo ed entra nei codici specifici del livello inferiore e così via;
- **secondo la dizione alfabetica** → questa modalità di navigazione può essere realizzata secondo due diverse logiche: quella per *"trigrammi"*, più complessa (il codificatore digita un testo ed il sistema ricerca ed estrae dal dizionario i testi che hanno in comune con quello digitato uno o più trigrammi) ed una seconda, *"per ordine alfabetico"*, (il codificatore digita un testo ed il sistema ricerca si posiziona nel dizionario nel corrispondente punto dell'ordine alfabetico);
- **secondo una procedura mista** → il codificatore seleziona un ramo della classificazione e quindi effettua una ricerca testuale nell'ambito di quel ramo.

5.2 Computing Platform

Blaise 4.6 supporta Windows 98 (e successive versioni) e Windows NT 4.0 (e successive versioni), su processori Pentium II o superiori.
La dimensione minima di memoria di sistema è di 32 Mb, anche se è preferibile una dimensione di almeno 64Mb.

5.3 La codifica assistita con Blaise

Come già accennato nel paragrafo 5.1, il *DATAMODEL* definisce la struttura dei file-dati in ambiente Blaise.
Uno step necessario, quindi, per costruire un'applicazione di codifica assistita è quello di caricare in un apposito *file-dati (DB)* la classificazione di riferimento.
È opportuno che tale classificazione sia memorizzata in un file ASCII.
Il **primo passaggio** è quello di creare in ambiente Blaise il *DATAMODEL* che accoglierà la classificazione e che dovrà quindi rappresentare la struttura della stessa.

La definizione del *DATAMODEL* deve essere effettuata in un file con estensione ***".bla"***; in questo file saranno anche esplicitate le modalità di accesso alla Db Blaise, ossia le modalità di navigazione nella classificazione che si intende rendere disponibili per il codificatore: secondo l'albero (gerarchico), alfabetica o per trigrammi (la modalità definita "mista" nel paragrafo 5.1 implica il rendere disponibile sia la navigazione ad albero che una di quelle che considera la sola dizione, sia esse per trigramma e/o alfabetica).
Come si vedrà di seguito, questo file avrà il nome che identifica la classificazione in ambiente Blaise (MetaName = nome del *DATAMODEL*)..
Questo passaggio precede necessariamente tutti gli altri in quanto questi necessitano di far riferimento al MetaName che identifica la classificazione.
Una volta creato il *DATAMODEL* che accoglierà la classificazione, il **secondo passaggio** è quello di caricare il file ASCII della classificazione nel *DATAMODEL* stesso.
Le istruzioni per il caricamento devono essere eseguite tramite un'apposita routine (*MANIPULA*), scrivendole in un corrispondente file con estensione ***".man"***. I file con questa estensione, contengono linee di codice per manipolare dati utilizzando l'apposito modulo di Blaise.
Il **terzo passaggio** è infine quello di attivare la funzione di codifica assistita, in corrispondenza della variabile da codificare, nel questionario elettronico (*DATAMODEL)* tramite cui vengono rilevati o caricati i dati della variabile oggetto di codifica.

Di seguito vengono esposti in dettaglio questi tre **passi** da seguire per la costruzione di un'applicazione di codifica assistita, riportando come esempio il caso della Classificazione delle attività economiche – ATECO; saranno evidenziate le differenze della procedura, a seconda che si intenda utilizzare o meno il "tramite" dell'albero.

1) Primo passaggio: creazione del *DATAMODEL* che accoglierà la classificazione

1.a) Senza il "tramite" dell'albero

È riportato nella figura 22 il programma con estensione ***".bla"*** che accoglierà la classificazione e che rappresenta quindi la struttura della stessa.

Figura 22 – Definizione DATAMODEL dizionario (ATECOdizionario - struttura)

```
DATAMODEL ATECOdizionario

SECONDARY
  Alfa = Descr                { campo su cui è definita la chiave necessaria se si vuole la ricerca alfabetica}
  Trig = Descr(TRIGRAM)       { campo su cui è definita la chiave necessaria se si vuole la ricerca per trigrammi}

FIELDS
  Code  "Codice come ATECO91" : STRING[8]
  Descr "Descrizione del codice" : STRING[200]
  Note  "Note"                : STRING[50]

RULES
  Code
  Descr
  Note

ENDMODEL
```

Di seguito si descrivono le sezioni che definiscono il ***DATAMODEL***:

- **SECONDARY** per questi *DATAMODEL* è necessario associare ad almeno un campo la funzione di chiave secondaria, in modo da poter successivamente effettuare la ricerca su questo campo; in questa sezione, quindi, si individua il campo testuale su cui sarà effettuata la ricerca e si associano ad esso uno o più criteri di ricerca (alfabetica o per trigrammi). Nell'esempio, il campo su cui effettuare la ricerca per chiave sarà ***"Descr",*** su cui potrà essere utilizzato sia il criterio di ricerca alfabetico (senza parola riservata) che quello per trigrammi (ciò è dichiarato con la parola riservata "TRIGRAM").
- **FIELDS** in questa sezione sono riportati i campi del ***DATAMODEL* Blaise** nei quali saranno caricati i campi dal file ASCII input (o parte di essi) della classificazione di riferimento. Nell'esempio, abbiamo i campi:

Code = codice
Descr = descrizione testuale
Note = eventuali note esplicative associate al record.

Indichiamo i vari "elementi" necessari per definire in Blaise ciascuna variabile:

field name - scritto per primo, riporta il nome della variabile utilizzato in Blaise
  - Code
  - Descr
  - Note

field text - scritto tra apici, contiene il nome "parlante" della variabile (tale testo è opzionale). Nell'esempio abbiamo:
  - *Codice come da ATECO91*
  - *Descrizione del codice*
  - *Note*

field type - dopo i 2 punti – riporta le caratteristiche della variabile (tipologia, lunghezza, etc.). Nei casi che stiamo trattando, nei quali il *DATAMODEL* è finalizzato a descrivere la struttura di una classificazione, avremo variabili tipo "Stringa", le cui

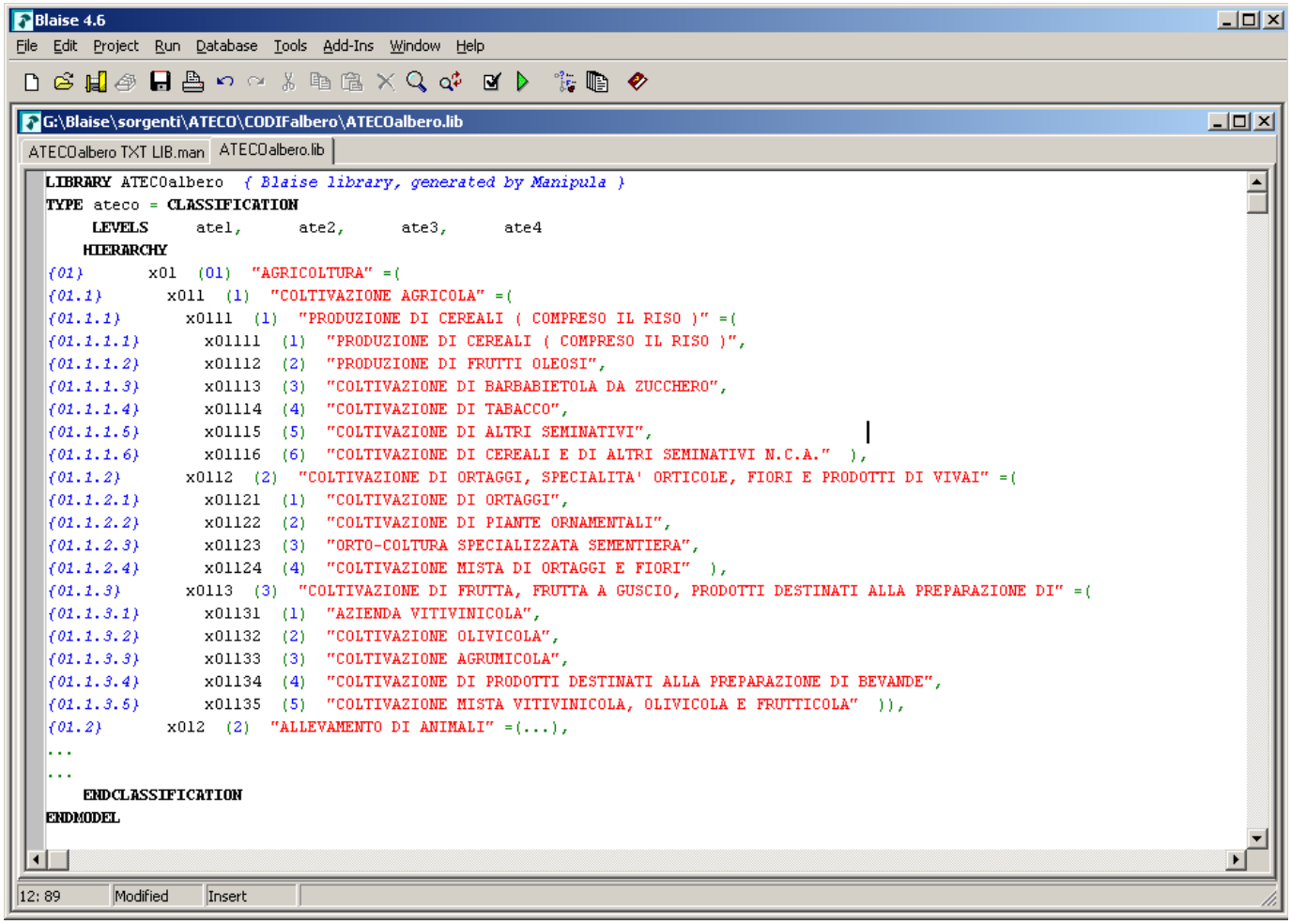
lunghezze potranno variare a seconda delle esigenze. Nell'esempio:

- STRING[8] → stringa di 8 caratteri
- STRING[200] → stringa di 200 caratteri
- STRING[50] → stringa di 50 caratteri

1.b) Con il "tramite" dell'albero

Volendo codificare tramite l'albero, la struttura gerarchica dei codici (l'albero) è implementata in un file con estensione ***".lib"***, cosiddetto ***"Library"***, con una sezione chiamata ***"Type".***
Come si può vedere dalla figura di sotto riportata, in questa sezione si enuncia che l'oggetto in esame è una classificazione (parola riservata = ***"Classification"***), si elencano i livelli gerarchici in base ai quali i codici sono strutturati (parola riservata = ***"Levels"***) e si riportano i "dati" della classificazione, preceduto dalla parola riservata ***"Hierarchy"***.

Figura 23 – Definizione TYPE classificazione (albero)



Come si vede dalla figura:

- **LIBRARY** individua il nome della libreria corrente (ATECOalbero), che descrive cioè l'albero;

- **TYPE** è la sezione in cui viene implementata la struttura ad albero della classificazione; come già detto, il fatto che tale sezione sia utilizzata per descrivere una classificazione è enunciato dalla parola riservata *"classification"*
- **LEVELS** riporta le etichette (labels) associate ai livelli gerarchici della classificazione;
- **HIERACHY** preannuncia che ciò che segue ossia è la descrizione della gerarchia della classificazione; in particolare:
  - racchiuso tra parentesi graffe → commento che racchiude il codice della classificazione, espresso con la simbologia desiderata (in questo caso, con i punti)
  - preceduto da una "x" → etichetta associata univocamente al codice
  - racchiuso tra parentesi tonde → progressivo del ramo nell'ambito del livello
  - descrizione testuale associata al codice, seguita da:
    - '=(' → se segue un sotto-livello
    - ',' → se il record che segue appartiene allo stesso livello
    - ')' → se il record in esame è l'ultimo di quel livello.

Per chiarire, rifacendoci all'esempio della figura 23, la nostra classificazione è costituita da 4 livelli gerarchici, rispettivamente *"ate1"*, *"ate2"*, *"ate3"* e *"ate4"*.
Il codice *"01"*appartiene al primo livello (*"ate1"*) , e la descrizione *"AGRICOLTURA"* è la prima di questo livello, che si estrinseca in più sottolivelli; il secondo sottolivello (*"ate2"*) è costituito da:

- codice *"011"*, descrizione *"COLTIVAZIONE AGRICOLA"* (primo ramo del secondo sottolivello)
- codice *"012"*, descrizione *"ALLEVAMENTO DI ANIMALI"* (secondo ramo del secondo sottolivello)
- ……..

È possibile generare questo file editandolo "a mano"; tuttavia, nel caso di classificazioni complesse come quella presa in esame, è possibile generarlo automaticamente con un qualsiasi linguaggio di programmazione, oppure con un apposito programma di estensione ***".man"*** (vedi ALLEGATO A)

In entrambi i casi, la ***"Library"*** deve essere resa disponile per il programma con estensione ***".bla"*** che accoglierà la classificazione; a tal fine deve essere eseguita la funzione ***"PREPARE"*** disponibile sulla barra comandi (control centre di Blaise) della schermata di EDIT.

La figura 24 rappresenta, per l'appunto, il programma ***".bla"*** che accoglierà la classificazione di cui è stata definita la struttura ad albero.
Come può vedersi, rispetto alla figura 1, abbiamo una sezione in più, ***LIBRARIES,*** in cui si richiama la ***"Library"*** sopra descritta.
Inoltre, in questo programma, il campo "Code" non è definito di tipologia stringa (come nella ***figura 1***), ma viene associato all'oggetto nominato "ATECO", definito precedentemente nella ***"library"*** di tipo ***"classification"***, cui corrisponde quindi la struttura ad albero lì definita.

Figura 24 – Definizione DATAMODEL dizionario (ATECOdizionario)

DATAMODEL ATECOdizionario { definizione struttura (file-dati) per navigazione **con albero** }

LIBRARIES
ATECOalbero *{ file ove è riportato l'albero della classificazione ATECO91}*

SECONDARY
Alfa = Descr { campo su cui è definita la chiave necessaria se si vuole la ricerca **alfabetica**}
Trig = Descr(TRIGRAM) { campo su cui è definita la chiave necessaria se si vuole la ricerca **per trigrammi**}

FIELDS
Code "Codice come da ALBERO" **: ATECO**
Descr "Descrizione del codice" **: STRING[200]**
Note "Note" **: STRING[50]**

RULES
Code
Descr
Note

ENDMODEL

Una volta definito il DATAMODEL che accoglierà il tipo classificazione, bisogna predisporre il DB al caricamento della stessa in ambiente Blaise, tramite le apposite funzioni di Blaise disponibili dalla schermata di edit (control centre di Blaise); tali funzioni devono essere eseguite in sequenza, procedendo alla seconda qualora la prima abbia dato esito positivo (trattasi delle funzioni ***"PREPARE"*** e ***"RUN"***).

2) Secondo passaggio: caricamento del file ASCII della classificazione nel *DATAMODEL* predisposto con il primo passaggio.

2.a) Senza il "tramite" dell'albero

È riportato nella figura 25 il programma con estensione ***".man"*** contenente le istruzioni per il caricamento del file esterno (formato ASCII) della classificazione ATECO.

Figura 25 – Caricamento DATAMODEL dizionario (da ASCII a BLAISE - **senza albero)**

```
USES
  DATAMODEL Importa
   FIELDS                          { descrizione dei campi del file di input da caricare }
    Codice   : STRING[8]
    Descr  : STRING[200]
    Note   : STRING[50]
   ENDMODEL

  ATECOdizionario  "ATECOdizionario"  { nome → METAinformazione – datamodel . bla  }

INPUTFILE                          { indirizzo del file di input e formato }
  Input: Importa("ATECOdizionario.csv", ASCII)
   SETTINGS
    SEPARATOR=';'

OUTPUTFILE                         { indirizzo del DB di output e formato }
  Output : ATECOdizionario        ("ATECOdizionario", BLAISE)

MANIPULATE                         { operazioni sul DB di output }
  WRITE(Output)
```

Di seguito sono descritte le varie sezioni del programma con estensione ***".man"***:

- **USES** si fa riferimento a due *DATAMODEL*
  il **primo** *DATAMODEL* citato è quello di input e riporta il tracciato record dei campi del file ASCII (chiamato nell'esempio *"importa")*;
  il **secondo** è quello di output ed individua il *DATAMODEL* Blaise in cui viene caricato il file di input (precedentemente descritto figura 22);
- **INPUTFILE** è riportato l'indirizzo del file ASCII da caricare ed il carattere utilizzato come delimitatore tra i campi
- **OUTPUTFILE** si indica il DB Blaise da scrivere a seguito del caricamento della classificazione
- **MANIPULATE** si esplicitano le "operazioni" di inserimento del file ASCII del DB Blaise (istruzione *"WRITE"*).

2.b) Con il "tramite" dell'albero

Come si può vedere dalla figura di seguito riportata, nella sezione "***MANIPULATE"***, è specificato il campo **"code"** che si aggancia alla sezione ***"TYPE"*** della ***"LIBRARY",*** in cui è stata definita la struttura ad albero della classificazione; su questo campo viene effettuata una funzione di assegnazione dei valori del campo "codice" della classificazione da caricare.

Figura 26 – Caricamento DATAMODEL dizionario (da ASCII a BLAISE - senza albero)

```
USES datamodel importa
  FIELDS
   Codice : STRING[8]
   Descr  : STRING[200]
   Note   : STRING[50]
 ENDMODEL
 ATECOdizionario 'ATECOdizionario'
```

```
INPUTFILE
 Input1: importa('ATECOdizionario.BLAISE. ASCII)
  SETTINGS
   SEPARATOR=';'
```

```
OUTPUTFILE
 Output1 : ATECOdizionario('ATECOdizionario', BLAISE)
```

```
MANIPULATE
 code := codice      { non è possibile attribuire alla var.(input e output) lo stesso nome perché sono di tipo diverso }
 WRITE(Output1)
```

Così come per il passaggio precedente, per eseguire questo passaggio è necessario eseguire le funzioni ***"PREPARE"*** e ***"RUN"***, disponibili dal pannello di Edit (control centre di BLAISE).

3) Terzo passaggio: attivazione della funzione di codifica nel questionario elettronico

Una volta predisposto l'ambiente di codifica (passaggi 1 e 2), affinché questo sia messo a disposizione dell'utente, è necessario che sia richiamato nell'ambito di un ***DATAMODEL*** che, come accennato nel paragrafo 5.1, potrebbe corrispondere all'intero questionario di rilevazione (qualora la codifica sia effettuata nel corso dell'intervista) oppure ad un sotto-insieme dello stesso (qualora i dati siano stati rilevati in altro modo e vengano caricati in tale modulo soltanto le variabili di interesse per la codifica).
Si riportano di seguito due ***DATAMODEL*** che fanno riferimento all'ambiente di codifica dell'Ateco, descritto nei primi due passaggi e che richiamano rispettivamente l'applicazione che non utilizza l'albero e quella che ne fa uso.

3.a) Senza il "tramite" dell'albero

Nella figura che segue è riportata la parte di questionario di intervista in cui si chiede al rispondente di descrivere la propria attività economica e viene attivata la funzione di codifica testuale, senza il tramite dell'albero.

Figura 27 – Esecuzione DATAMODEL CODIFICA senza albero

DATAMODEL ATECOquestionario "Indagine codifica ATECO91 - senza albero"

```
USES                    {definisce il  nome del DATAMODEL esterno da richiamare (metaInformazione) }
  B 'ATECOdizionario'
```

```
EXTERNALS               {individua il/i file BLAISE esterno/i da utilizzare}
  Ext: B ('ATECOdizionario') {associazione: campo esterno → metaInformazione (in uses)}
```

```
FIELDS                  { campi costituenti il questionario elettronico in cui viene effettuata la codifica }
  Testo  "@HPuò descrivere la sua attività economica? @H.
        @/Per l'intervisatore: digitare la risposta fornita, quindi PREMERE il tasto INVIO "      : STRING[100]
  Codice "PREMERE la BARRA SPAZIATRICE e, una volta individuato il codice corrispondente
          @/ alla risposta fornita, selezionare il box  SELECT@."                                  :
STRING[8]
RULES                   { sequenza ed operazioni eseguite nell'ambito del DATA MODEL }
  Testo
  Codice | Ext.LOOKUP(Trig:=(Testo),Alfa:=(Testo)).code
```

ENDMODEL

Come si può vedere, il programma è costituito da quattro sezioni ed, in particolare:

- **USES** individua il ***DATAMODEL*** che rappresenta la classificazione da utilizzare (**metaInformazione)** e lo associa ad una Label (B) che viene richiamata nella sezione successiva;
- **EXTERNALS** individua il file Blaise esterno contenente la classificazione da utilizzare associandolo alla label definita nella sezione precedente;
- **FIELDS** elenca i campi nei quali vengono registrate le variabili rilevate in corso di intervista. Nell'esempio abbiamo:

  Testo = campo, di tipo stringa, in cui registrare la descrizione della attività economica del rispondente; in particolare, il testo tra apici contiene il wording del quesito e le istruzioni per il rilevatore/codificatore

  Codice = campo, di tipo stringa, in cui viene memorizzato il codice corrispondente alla descrizione fornita, utilizzata per la ricerca testuale (**LOOKUP**) secondo i vari criteri

- **RULES** indica la sequenza con cui è effettuato il display delle variabili a video (in base alla quale quindi vengono posti i quesiti nel corso dell'intervista). In particolare, viene prima posto il quesito in cui si chiede di descrivere l'attività economica (Testo) e poi viene effettuato il display della variabile in cui sarà registrato il corrispondente codice (Codice). In corrispondenza di questo campo Codice, viene richiamata la funzione di codifica assistita, tramite l'istruzione

| ExtB.LOOKUP(Trig:=(Testo), Alfa:=(Testo)).code in cui:

ExtB → richiama il file esterno contenente la classificazione
LOOKUP → parola riservata che attiva la funzione di ricerca testuale (matching)
Trig → rende disponibile il matching per trigrammi
Alfa → rende disponibile la ricerca alfabetica
Code → campo che restituisce il codice.

3.b) Con il "tramite" dell'albero

Nella figura che segue è riportata la parte di questionario di intervista in cui si chiede al rispondente di descrivere la propria attività economica e viene attivata la funzione di codifica con il tramite dell'albero.

Figura 28 – Esecuzione DATAMODEL CODIFICA con albero

DATAMODEL ATECOquestionario "Indagine codifica ATECO91 - con albero"

```
LIBRARIES          { libreria usata – file albero della classificazione }
  ATECOalbero 'ATECOalbero'

USES               { elenco metaInformazione → file usati nell'ambito del DATA MODEL (associazione) }
  A 'ATECOdizionario'

EXTERNALS          {elenco file esterni utilizzati in un blocco (path se non stesso ambiente – associazione)}
  ExtA: A ('ATECOdizionario')        {associazione: campo esterno → metaInformazione (in uses)}

FIELDS             { campi costituenti il questionario elettronico in cui viene effettuata la codifica }
  Testo  "@HPuò descrivere la sua attività economica? @H.
        @/Per l'intervisatore: digitare la risposta fornita, quindi PREMERE il tasto INVIO "      : STRING[100]
  Codice "PREMERE la BARRA SPAZIATRICE e, una volta individuato il codice corrispondente
          @/ alla risposta fornita, selezionare il box  SELECT@."                                  : ATECO   { tipo
classificazione }

RULES              { sequenza ed operazioni eseguite nell'ambito del DATA MODEL }
     Testo
     Codice.classify  | ExtA.LOOKUP(Trig:=(Testo),Alfa:=(Testo)).code

ENDMODEL
```

Rispetto a quanto riportato nella figura 6 (attivazione della funzione di codifica senza l'albero), in questo caso il *DATAMODEL* è costituito da una sezione in più:

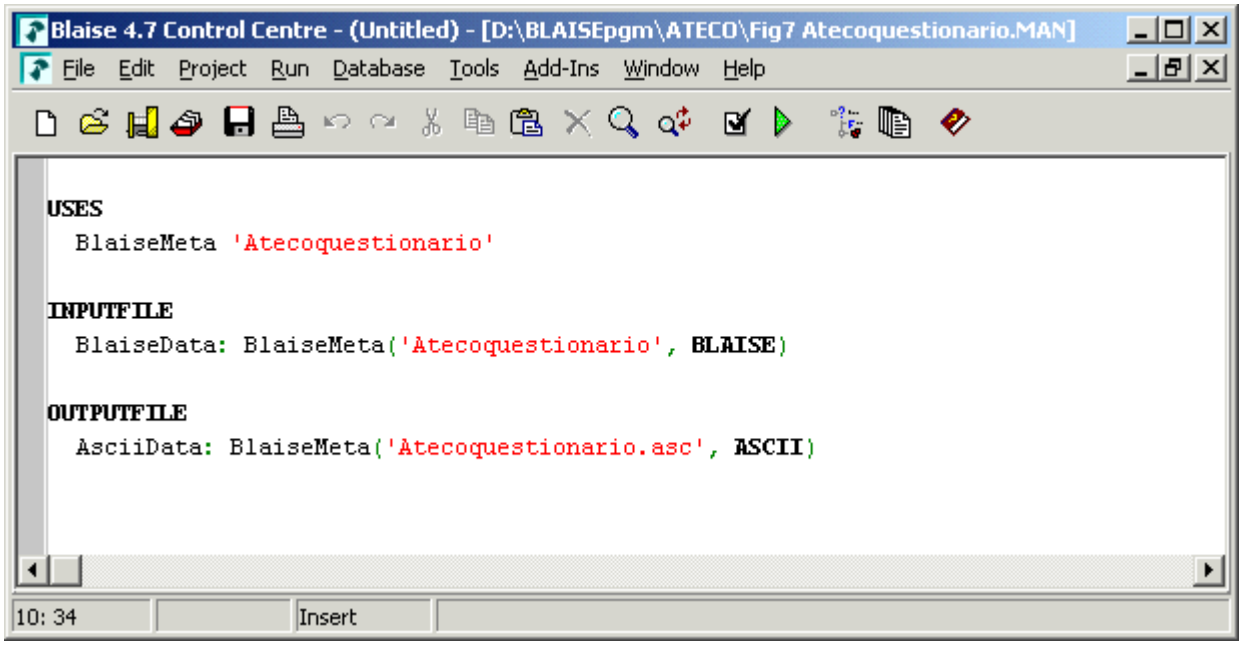
- **LIBRARIES** richiama il nome del DATAMODEL (ATECOalbero), che descrive cioè l'albero della classificazione).

Altre due differenze sono riscontrabili nelle sezioni FIELDS e RULES e, rispettivamente:

- nella sezione FIELDS il campo Codice non è definito di tipologia stringa (come nella figura 27), ma viene associato all'oggetto nominato "ATECO", già definito nell'apposita ***"library"*** di tipo ***"classification"***, cui corrisponde quindi la struttura ad albero lì definita.
- nella sezione RULES con l'espressione ***Codice.classify*** si attesta che il codice deve essere attribuito secondo la classificazione la cui struttura ad albero è descritta nella sezione ***LIBRARIES***.

Una volta codificati i dati eseguendo il programma ***".bla"*** (sia che si sia codificato con o senza il tramite dell'albero), avremo che i dati saranno registrati su un DB (formato Blaise) avente lo stesso nome del file ***".bla"*** ma estensione ***".bdb"***.
Per poterli trasformare in formato ASCII per essere elaborati con qualsiasi software, è sufficiente eseguire un programma manipula con le istruzioni riportate nella seguente figura.

Figura 29 – Programma MANIPULA per lo scarico dei dati da BLAISE ad ASCII



È opportuno sottolineare che finora abbiamo distinto le applicazioni di codifica ***"con"*** e ***"senza"*** l'albero, assumendo che si voglia comunque mettere a disposizione dell'utente la funzione di ricerca testuale.

Tuttavia, è possibile arrivare all'attribuzione del codice navigando nell'albero della classificazione senza effettuare alcuna ricerca testuale.

In questo caso:

- non c'è bisogno di registrare il testo (campo Testo)
- non c'è bisogno di caricare in ambiente Blaise il file ASCII della classificazione (codice + testo), ma è sufficiente implementarne la struttura ad albero nell'apposita ***Library***
- nella sezione ***Rules*** del programma *".bla"* per l'attivazione della funzione di codifica, è sufficiente scrivere l'istruzione Codice.classify.

La figura che segue riporta il programma per la codifica tramite l'albero, senza la ricerca testuale.

Figura 30 –DATAMODEL per la codifica tramite l'albero, senza la ricerca testuale

DATAMODEL ATECOquestionario "Indagine codifica ATECO91 - con albero"

LIBRARIES { libreria usata (TYPE) – file albero della classificazione }
ATECOalbero 'ATECOalbero'

FIELDS { campi costituenti il questionario elettronico in cui viene effettuata la codifica}
Codice "PREMERE la BARRA SPAZIATRICE e, una volta individuato il codice corrispondente
@/ alla risposta fornita, selezionare il box SELECT@." **: ATECO** { tipo classificazione }

RULES { sequenza ed operazioni eseguite nell'ambito del **DATA MODEL** }
……
Codice.classify

ENDMODEL

Per completezza si riportano di seguito i principali TIPI di file Blaise (estensioni) utilizzati e generati dalle applicazioni di codifica assistita.

File sorgenti (originari)

Estensione	Descrizione	Tipo
BLA	Definizione della struttura dei file-dati in ambiente Blaise	ASCII
LIB	Libreria in cui vengono definiti i formati associati a campi del DATAMODEL	ASCII
MAN	Sorgente che consente la manipolazione dei dati	ASCII

File generati dal modulo ".bla"

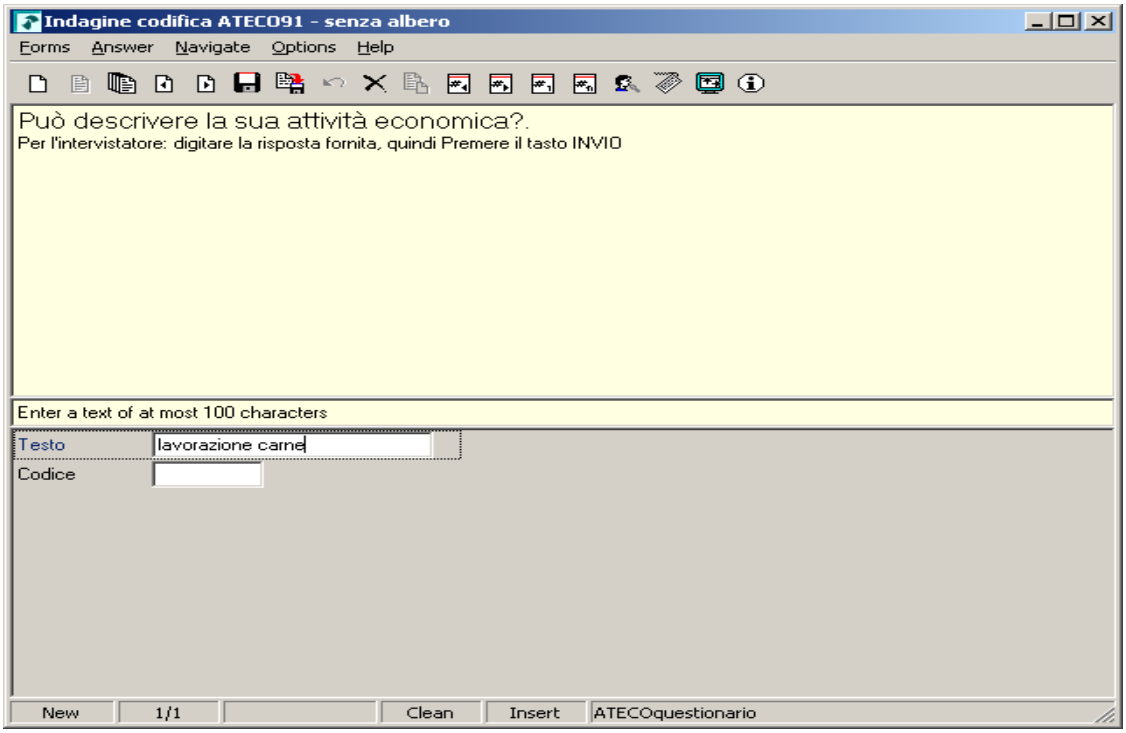
Estensione	Descrizione	Tipo	Visualizza (EDITOR)	Note
BDB	Dati Blaise.	BINARIO	Sistema Blaise	File dati in ambiente Blaise.
BDT	Trigrammi del sistema Blaise.	BINARIO		Contiene tutti i trigrammi. È generato se è definita una chiave secondaria di tipo trigramma nel data-model.
BFI	Informazioni di sistema sui file Blaise.	BINARIO		Informazioni interne circa i file Blaise generato quando vengono creati i file dati.
BIT	Blaise Indice sui trigrammi.	BINARIO		Gli indici del file BDT (trigrammi). È' creato se una chiave secondaria di tipo trigramma è dichiarata nel data-model.
BJK	File di indici interno al sistema Blaise	BINARIO		Indice generato dal sistema Blaise quando i file dati (DBD) sono creati inizialmente.
BMI	struttura dei file-dati in	BINARIO	Sistema Blaise	È generato quando si lancia il "PREPARE" di un data-model.

	ambiente Blaise			
BSK	File di indici generati sulla chiave secondaria.	BINARIO		Indici definiti sulla chiave secondaria. È generato se una chiave secondaria è dichiarata nel data-model.

5.4 L'interfaccia utente della funzione di codifica

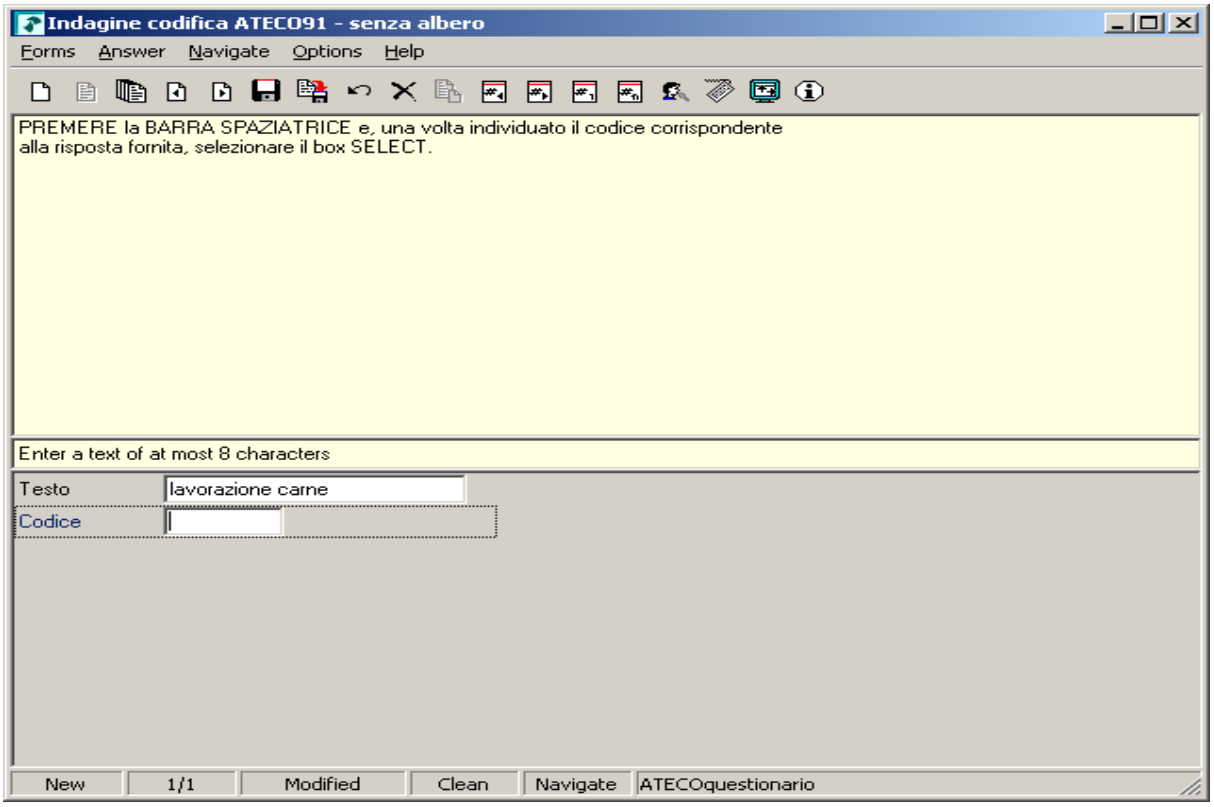
Sono riportate di seguito le schermate proposte al codificatore (o intervistatore, se la codifica viene effettuata in corso di rilevazione), a seconda che sia stata predisposta l'applicazione per la codifica testuale, ad albero o mista.
Nel caso della codifica in base al testo, la prima schermata sarà quella in cui deve essere digitato il testo da codificare (nel caso la codifica sia effettuata in una fase successiva rispetto all'intervista, potrà essere evitata la digitazione delle risposte già rilevate, caricandole nel DATAMODEL predisposto per la codifica, come già accennato nel par. 5.1).

Figura 31 - Schermata per l'acquisizione del testo da codificare



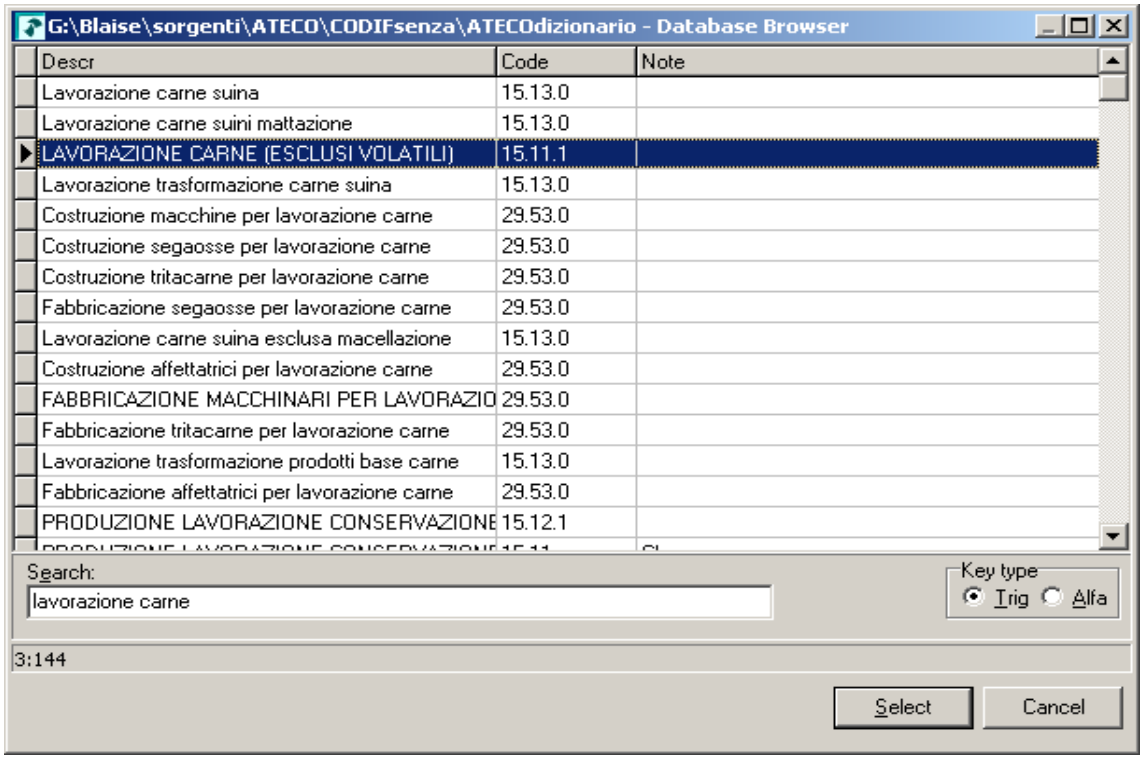
A seguito dell'invio, il cursore si sposta sul campo in cui sarà successivamente registrato il codice e, per attivare la funzione di codifica ed accedere quindi alla navigazione nel dizionario, sarà richiesto al codificatore di premere la barra spaziatrice figura 32.

Figura 32 - Schermata per l'attivazione della funzione di codifica



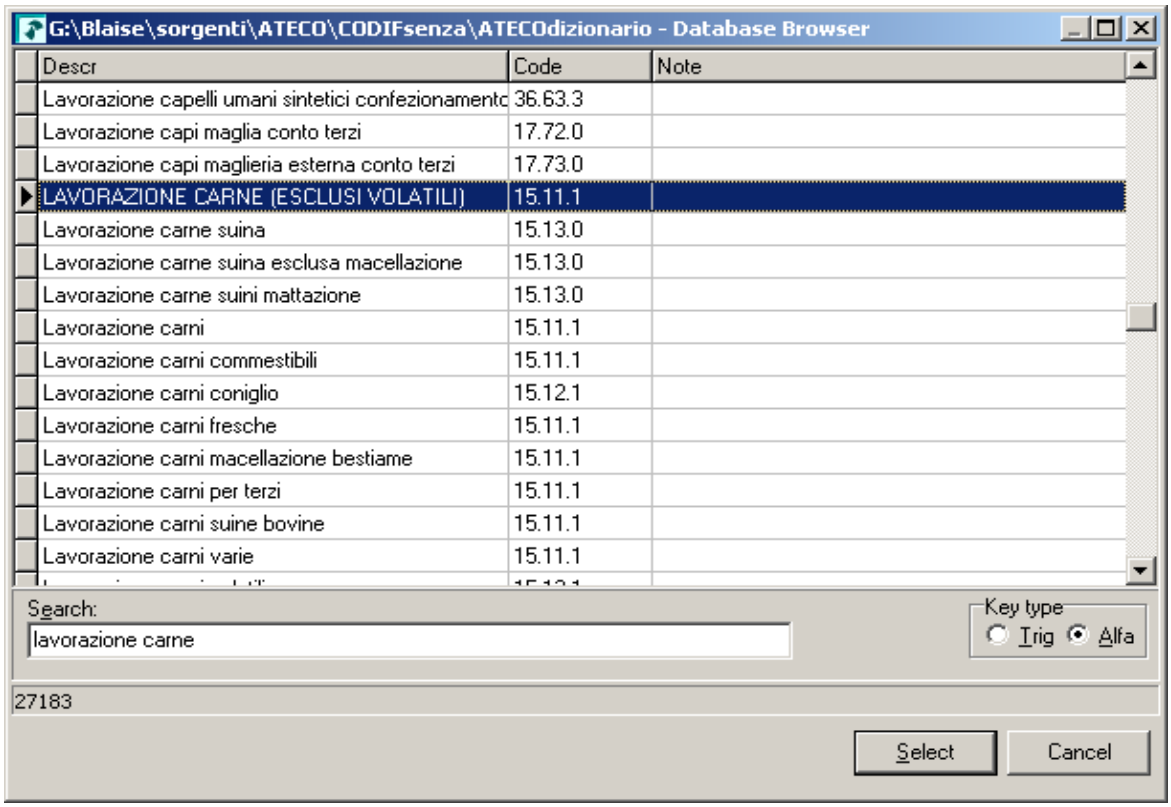
La schermata successiva contiene il display dei testi estratti dal dizionario a seguito del matching in base ai trigrammi, perché questa era la prima opzione citata nel programma (confrontare figura 27).

Figura 33 - Schermata per la navigazione nel dizionario in base ai trigrammi



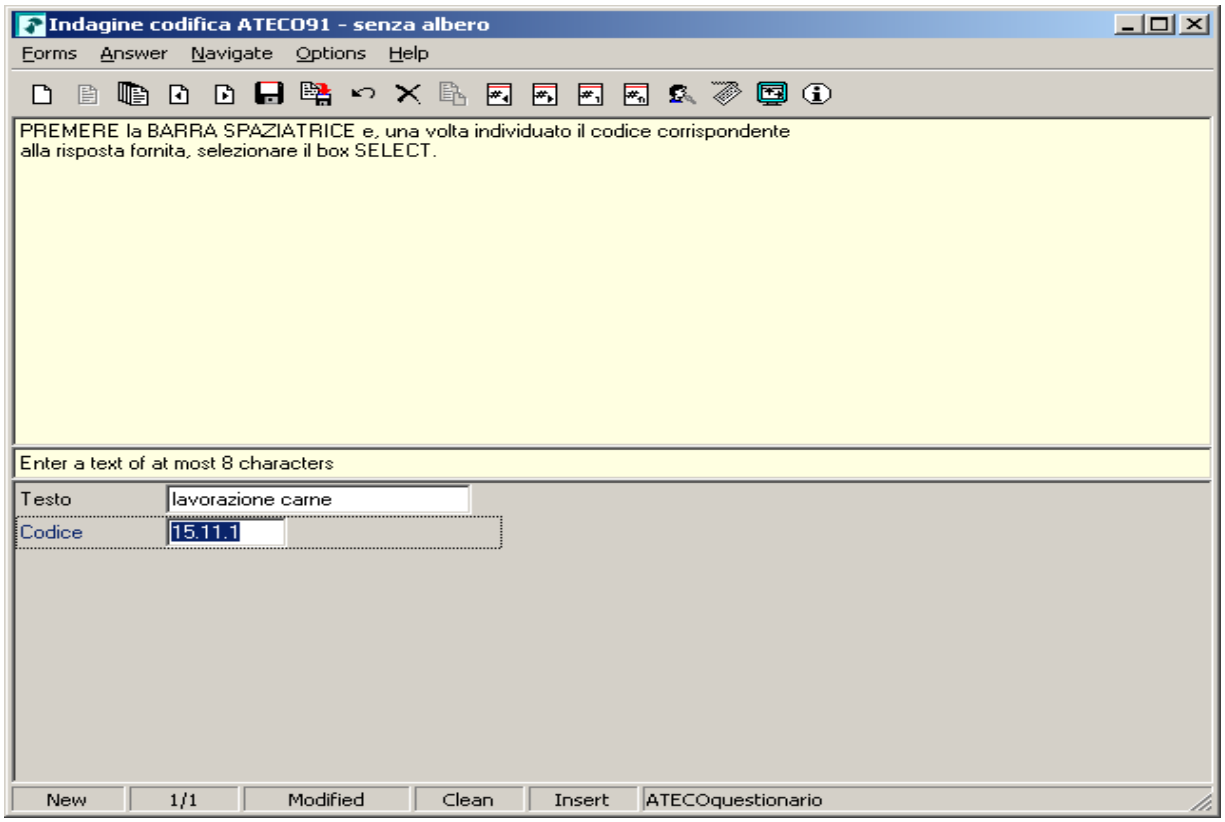
Qualora l'intervistatore voglia optare per la ricerca alfabetica, dovrà cliccare nell'apposito box; la schermata che gli verrà proposta in base a questo criterio di ricerca sarebbe la seguente.

Figura 34 Schermata per la navigazione nel dizionario in base alla ricerca alfabetica



Il codificatore selezionerà il testo del dizionario corrispondente alla risposta da codificare, muovendosi con il cursore; quindi, premendo INVIO, sul box SELECT, il corrispondente codice sarà registrato nell'apposito campo del *DATAMODEL* corrente (confrontare figura 35).

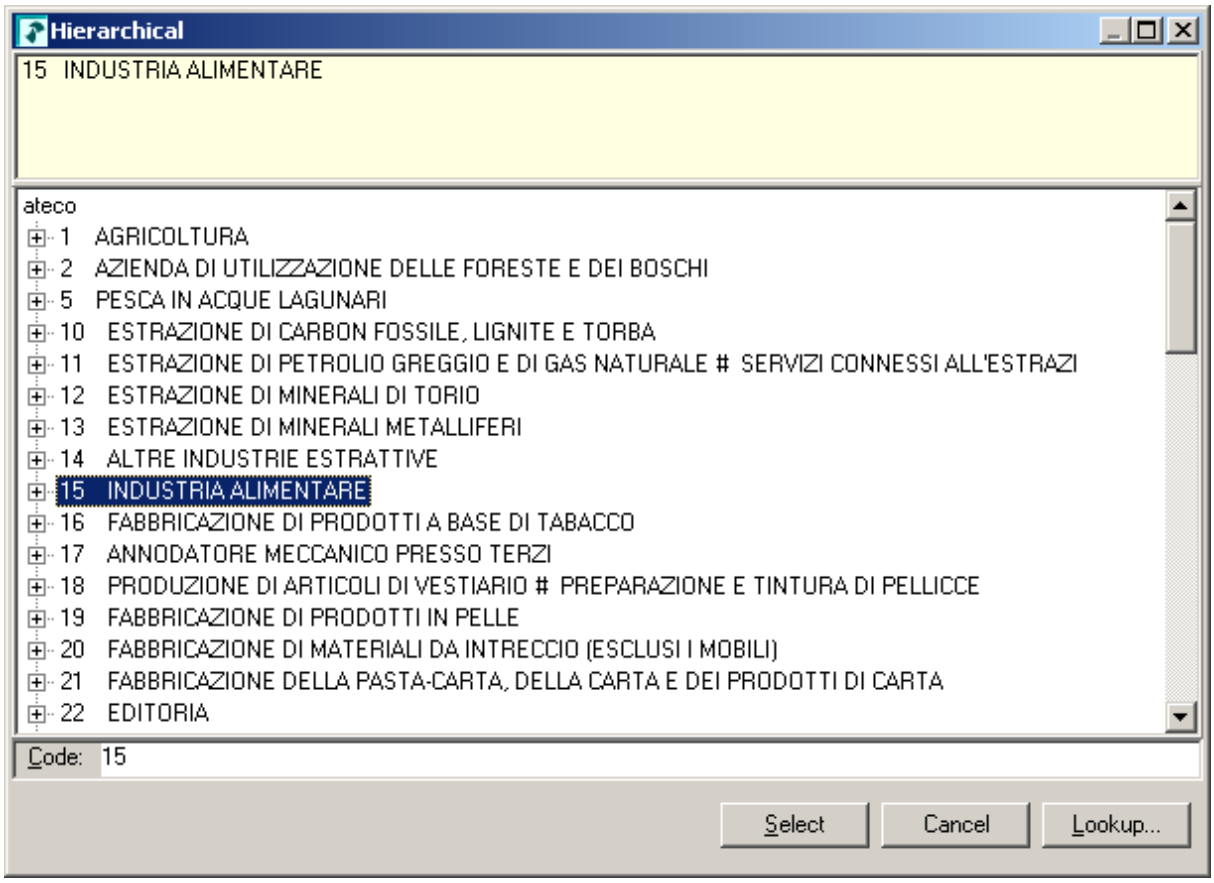
Figura 35 - Schermata di ritorno a seguito della selezione del codice



Nel caso sia stata predisposta un'applicazione per la codifica mista, dopo avere digitato il testo da codificare e dato invio (confrontare figura 32***)***, sarà proposta la seguente schermata, in cui viene effettuato il display della radice (rami gerarchicamente superiori) della classificazione.
Movendosi con il cursore, il codificatore selezionerà il ramo corrispondente alla risposta da codificare e, dando invio su quello selezionato, potrà eventualmente accedere al ramo gerarchicamente inferiore.
Una volta selezionato il ramo di interesse, il codificatore ha due possibilità:

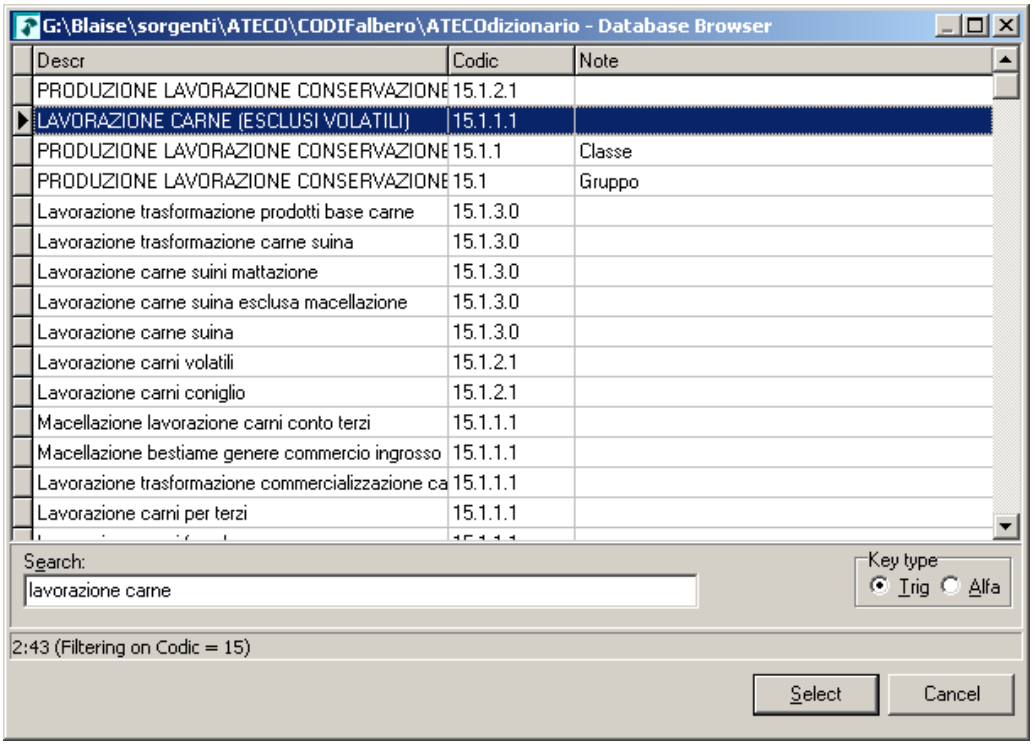
- dare INVIO sul box SELECT → così sarà registrato sul record il codice corrispondente al ramo
- dare INVIO sul box LOOKUP → così attiverà la funzione di codifica in base al testo, limitando il matching del testo acquisito nell'ambito dei record del dizionario appartenenti al ramo selezionato.

Figura 36 - Selezione del ramo della classificazione



Come nel caso precedente, il codificatore si muoverà con il cursore per selezione il testo del dizionario corrispondente alla risposta da codificare; quindi, premendo INVIO sul box SELECT, il corrispondente codice sarà registrato nell'apposito campo del *DATAMODEL* corrente.

Figura 37 - Selezione del ramo della classificazione



Nel caso sia stata attivata la funzione di codifica esclusivamente ad albero, non sarebbe stato necessario acquisire e quindi digitare la risposta testuale, quindi, dando INVIO sul campo CODICE, sarebbe stata proposta la schermata della figura 36 nella quale, però, non sarebbe stato disponibile il box LOOKUP.

6 Guida all'utilizzo di Blaise per la codifica assistita

Finora sono stati descritti gli aspetti tecnici per costruire un'applicazione di codifica in Blaise; passiamo ora a fornire una serie di spunti, derivanti soprattutto dall'esperienza acquisita, che possono essere utili per ottenere prestazioni migliori.
Innanzi tutto è opportuno distinguere se l'applicazione da implementare si basi su una classificazione *"semplice"* oppure *"complessa"*. È bene allora chiarire da che punto di vista si intenda affermare che una classificazione abbia l'una o l'altra delle caratteristiche citate.

Possiamo dire che una classificazione è "*semplice*" se:

- il sistema di codici è strutturato in un numero limitato di livelli;
- la discriminazione tra i rami dello stesso livello è facilmente individuabile;
- le definizioni della classificazione ufficiale sono abbastanza brevi;
- le definizioni della classificazione ufficiale non danno adito alla necessità di inserire nel dizionario elaborabile un numero elevato di sinonimi;
- la variabilità del linguaggio utilizzato dai rispondenti non è eccessiva.

Una classificazione di questo genere può essere, per esempio, quella dei Comuni, relativamente alla quale:

- l'articolazione gerarchica è su tre livelli (regione, provincia, comune);
- non c'è ambiguità interpretativa tra i rami dello stesso livello;
- i nomi di regioni, province e comuni sono mediamente abbastanza brevi;
- la necessità di aggiungere sinonimi si ravvisa soprattutto al livello dei nomi dei comuni (per esempio, nell'applicazione di codifica implementata in Istat, sono stati inseriti nel dizionario, tra l'altro, i nomi delle "località" di pertinenza di ciascun comune);
- la variabilità di linguaggio utilizzato dai rispondenti è abbastanza limitata.

Nel caso in cui si voglia costruire un'applicazione che si basi su una classificazione con queste caratteristiche, non si ravvisa la necessità di fornire ulteriori elementi, oltre a quelli tecnici già descritti. Diverso è il caso di classificazioni più *complesse*, come, per esempio, quella della Professione, in particolare la Classificazione Istat 2001 che riprende sostanzialmente quella internazionale della Isco 1988.

Questa classificazione è infatti caratterizzata da:

- un'articolazione gerarchica su 4 livelli: 9 Grandi Gruppi articolati a loro volta in Gruppi, quindi Classi nei Gruppi e Categorie nelle Classi. In particolare, i Grandi Gruppi sono 9, i Gruppi sono 37, le Classi sono 121 e le Categorie sono 519, in corrispondenza delle quali sono state riconosciute 6300 Voci professionali, alle quali deve essere ricondotto l'universo non quantificabile delle professioni;
- la discriminazione tra i rami dello stesso livello non è sempre così immediata, infatti la logica della Classificazione ufficiale è basata sul criterio della competenza, intesa nella sua duplice dimensione: del livello delle competenze e del campo di applicazione delle competenze. La prima dimensione coglie una differenza gerarchica tra le professioni ed è assimilabile, sostanzialmente, al livello di istruzione formale necessario allo svolgimento di una data professione. La seconda,

invece, consente una articolazione orizzontale delle professioni e viene usata principalmente per cogliere le differenze interne ai Grandi Gruppi in relazione alle differenze settoriali, alle attrezzature utilizzate, ai materiali lavorati, alla natura dei servizi prodotti ed a altre caratteristiche specifiche dell'ambito in cui si svolgono le diverse professioni. Altri criteri sottesi dalla Classificazione sono:

- il livello di *responsabilità;*
- il grado di *autonomia;*
- la *complessità* del lavoro;
- la componente *manuale/intellettuale* delle mansioni;

- le definizioni della Classificazione ufficiale non sono sempre brevi;
- la complessità della variabile in osservazione è tale che le definizioni della Classificazione ufficiale sottendono spesso una serie di concetti che rendono indispensabile l'inserimento nel dizionario elaborabile di un numero elevato di sinonimi;
- la variabilità di linguaggio utilizzato dai rispondenti è molto ampia, in quanto è facilmente immaginabile come una stessa professione possa essere descritta in diversi modi.

Si pensi che l'applicazione di codifica implementata in Blaise per la variabile Professione si compone di:

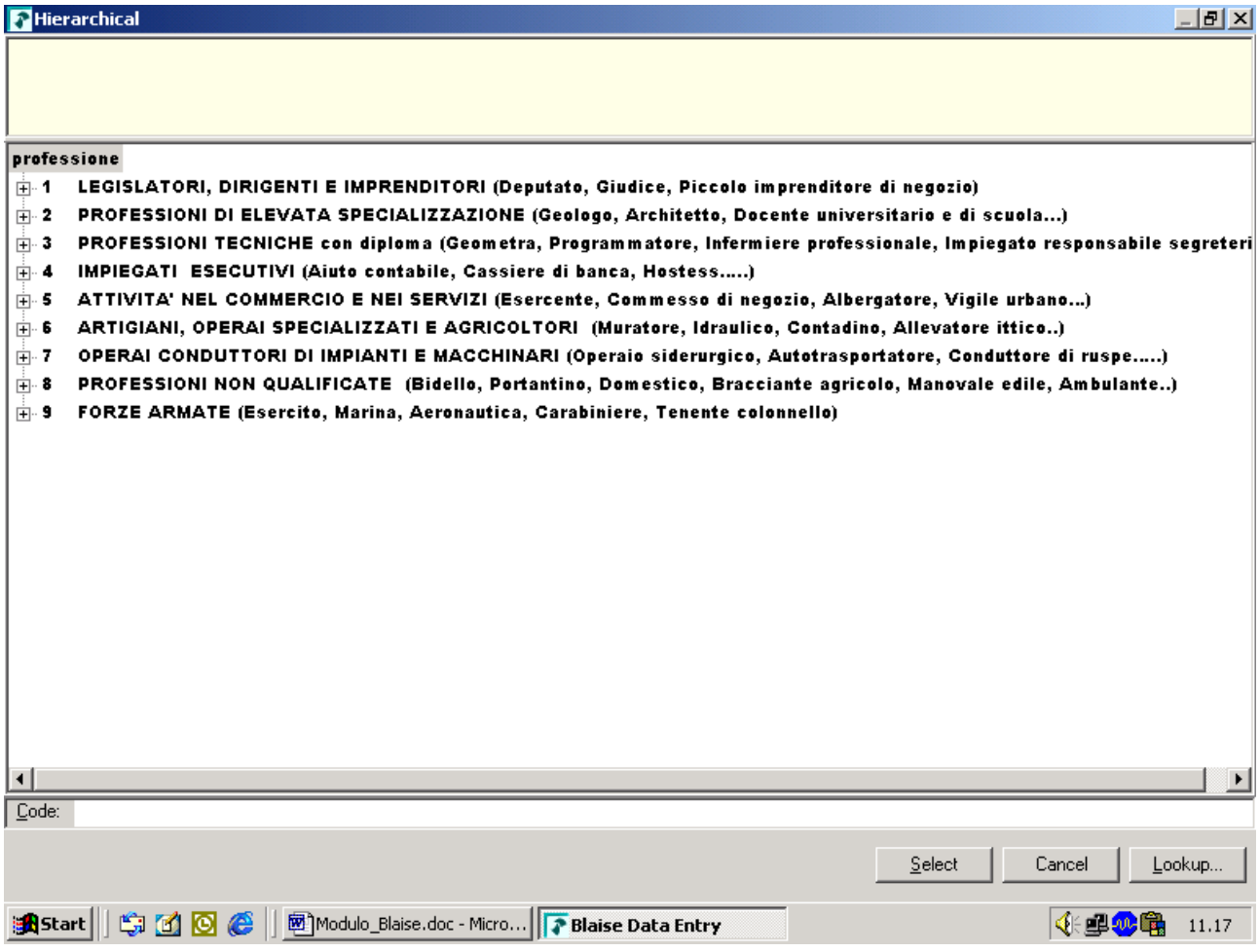
- un file ***Library*** che costituisce l'albero della classificazione (confrontare paragrafo 5.3) costituito da **686** record, al cui interno sono presenti solo **testi unici** associati a codici univoci completi e non (uguali o minori di quattro digit);
- un dizionario elaborabile (***lookup file***) costituito da **16.168** record, al cui interno sono presenti sia lemmi corrispondenti a codici a quattro digit (massimo dettaglio) che testi associati a codici con un numero inferiore di digit (codici generici), in quanto le informazioni contenute in essi non consentono di arrivare al massimo dettaglio; l'arricchimento di tale dizionario è stato effettuato principalmente avvalendosi di risposte empiriche pre-codificate e validate da esperti della classificazione, rilevate nel corso delle seguenti indagini:
  - *Indagine di qualità del Censimento della popolazione 1991*
  - *Indagine sulla salute 1994*
  - *Indagine sulle Forze di lavoro* 4 trimestri 1998
  - *I e II Indagine pilota del Censimento della popolazione 2001*
  - *Censimento popolazione 2001* (modello relativo alle Convivenze).

Costruire un'applicazione di questo genere ha richiesto una serie di accorgimenti, indubbiamente estendibili ad altre applicazioni che si basino su classificazioni altrettanto *"complesse"*.
Questi accorgimenti sono in parte associabili ad aspetti contenutistici da esplicitare per fornire un aiuto guidato al codificatore, in parte dovuti ad aspetti tecnici inerenti gli algoritmi utilizzati da Blaise per il *matching* testuale e, da ultimo, inerenti la necessità di verificare la qualità, in termini di *"efficacia"* ed *"accuratezza"* dell'applicazione sviluppata.

6.1 Accorgimenti per meglio “guidare” il codificatore

Una delle necessità ravvisate nella costruzione di un’applicazione complessa come quella della Professione è stata quella di limitare la possibilità di errore del codificatore nel selezionare i rami gerarchicamente superiori.
Si fa infatti presente che, nel caso di ricerca secondo l’albero della Classificazione, l’iniziale errata individuazione del Grande Gruppo (uguale al 1° digit del codice), effettuata da codificatori poco esperti, può compromettere definitivamente l’esito della successiva ricerca.
A prescindere quindi dal fatto che è stato giudicato consigliabile l’utilizzo del metodo di ricerca ad albero solo per personale che abbia una conoscenza approfondita della logica su cui si basa la Classificazione ufficiale, è stato ritenuto opportuno associare alle descrizioni ufficiali dei Grandi Gruppi del file contenente la struttura ad albero della classificazione (***Library)***, degli ***Esempi*** di professioni che ricadono all’interno del Grande Gruppo (racchiusi tra parentesi tonde). Negli esempi si è volutamente scelto di non fare esclusivamente riferimento alle Voci professionali riportate all’interno del manuale ufficiale, bensì di inserire professioni, non solo più vicine al linguaggio utilizzato dai rispondenti, ma anche e soprattutto che contengono le specifiche relative alle informazioni di dettaglio necessarie per arrivare ad una codifica corretta ed univoca (ad esempio *“Piccolo imprenditore di negozio”, “Impiegato responsabile segreteria”, “Bracciante agricolo”..).*
Nella figura che segue è riportata la schermata dell’applicazione di codifica in Blaise in cui viene effettuato il display dei rami gerarchicamente superiori della classificazione (Grandi Gruppi), corredatati dai citati esempi.

Figura 38 - Grandi Gruppi della Classificazione 2001: esempi inseriti nel file Library



Un altro aspetto rilevante è stato quello inerente l'utilizzo delle ***Note***, riportate in appositi campi associati alle descrizioni del dizionario elaborabile (*lookup file*), che sono state utilizzate principalmente per dirimere l'ambiguità della codifica di professioni generiche non codificabili univocamente.
Nel dizionario elaborabile, infatti, sono presenti anche testi relativi a professioni generiche, passibili cioè di essere codificate anche all'interno di Grandi Gruppi diversi ed, a volte, tali **testi** possono essere **uguali tra di loro, ma associati a codici diversi**, anche dal primo digit.
I ***Campi Note*** sono stati quindi inseriti a fianco dei testi corrispondenti a codici non completi e soprattutto alla tipologia di testi appena descritta. Le funzioni di questi ***Campi Note*** sono allora quelle di:

- avvisare l'operatore che si sta rilevando una professione generica;
- segnalare la necessità di chiedere all'intervistato ulteriori specifiche sulla professione dichiarata, per arrivare ad una codifica al massimo dettaglio, qualora si debba procedere ad effettuare la codifica in corso d'intervista;
- porre una maggiore attenzione nell'attribuzione dei digit mancanti, qualora l'operatore stia effettuando una codifica assistita post intervista.

Si veda la tabella che segue, per comprendere meglio la casistica appena descritta.

Tabella 1 - Esempio di utilizzo del campo note nel dizionario elaborabile

Codice	***Testo***	***Campo note***
6.4.1	*OPERAIO AGRICOLO*	*Operai agricoli specializzati generici*
7	*OPERAIO AGRICOLO*	*Operai agricoli conduttori impianti macchinari generici*

Nell'esempio riportato, la presenza del ***Campo Note*** segnala all'operatore che la professione corrispondente alla descrizione *"operaio agricolo"* è prevista in due Grandi Gruppi diversi e che pertanto è necessario conoscere altre informazioni per arrivare ad una codifica corretta e univoca: è *"un operaio agricolo specializzato?"*, oppure è *"un conduttore di impianti e macchinari agricoli?"*.

Oltre all'inserimento delle *Note*, un altro accorgimento per minimizzare gli errori di codifica è stato quello relativo all'introduzione, nei testi delle empiriche inserite nel dizionario informatizzato, delle cosiddette ***Stringhe/Parole chiave***.
L'utilizzo delle ***parole chiave*** è opportuno soprattutto se l'attività di codifica viene effettuata nel corso dell'intervista; in tal caso, infatti, chiarire con il rispondente il contenuto della propria Professione in modo da ricondurlo a ***parole chiave*** appositamente contemplate nel dizionario elaborabile consente all'operatore di discriminare più facilmente all'interno dei nove Grandi Gruppi.
Per capire come si è arrivati a determinare ed individuare le s*tringhe/parole chiave*, consideriamo come esempio la codifica delle "*Professioni impiegatizie"*, che la Classificazione ufficiale pone all'interno di due Grandi Gruppi: il 3 (*Professioni tecniche*) ed il 4 (*Impiegati*). In base ai criteri classificatori validati dai responsabili della Classificazione ufficiale, sono stati considerati sinonimi di:

- *professioni tecniche* (Grande Gruppo 3)-→ gli i*mpiegati direttivi, gli impiegati con funzioni di responsabilità o di coordinamento;*
- *impiegati* (Grande Gruppo 4) -→ gli *impiegati esecutivi, gli impiegati con mansioni generiche o di concetto;*

In questo caso *stringhe/parole chiave* diventano:

- *impiegato tecnico → impiegato direttivo → impiegato responsabile → impiegato coordinatore;*
- *impiegato → impiegato esecutivo → impiegato di concetto → impiegato mansioni generiche*

La tabella che segue evidenzia le altre *Stringhe/Parole chiave* che sono state contemplate nei testi delle empiriche del dizionario informatizzato e i Grandi Gruppi/Gruppi all'interno dei quali esse permettono di discriminare meglio la ricerca

Tabella 2 – Elenco delle Stringhe/Parole chiave e Grande Gruppo/Gruppo corrispondente

Stringhe/Parole chiave	***Grande Gruppo/Gruppi***
Dirigente pubblico/privato	Gruppi 1.1 e 1.2
Imprenditore grande/piccola azienda privata	Gruppi 1.2 e 1.3
Gestore/titolare	Gruppo 1.3
Specialista	Grande Gruppo 2
Tecnico/impiegato direttivo/impiegato responsabile/impiegato coordinatore	Grande Gruppo 3 e Gruppo 3.3
Impiegato esecutivo/impiegato di concetto/impiegato mansioni generiche	Grande Gruppo 4
Operaio specializzato/artigiano	Grande Gruppo 6
Operaio conduttore impianti industriali/operaio addetto macchinari/operaio lavorazioni in serie/operaio catena montaggio	Grande Gruppo 7
Operaio generico/operaio non qualificato/manovale/bracciante/operaio semplice/salariato	Grande Gruppo 8
Militare	Grande Gruppo 9

6.2 Accorgimenti di tipo tecnico

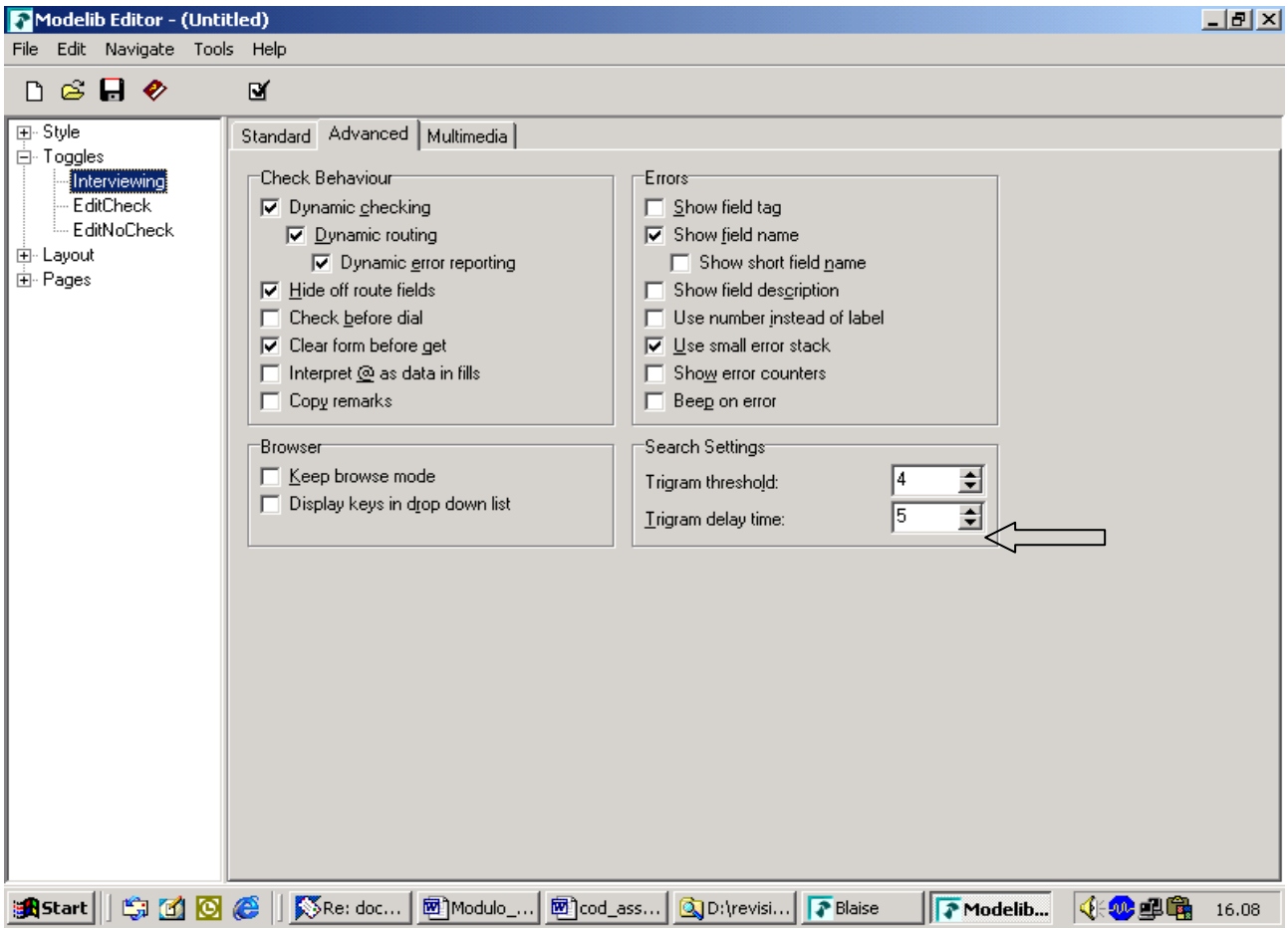
Come descritto nel paragrafo 1.4.2.2., la fase di *matching* testuale viene realizzata in Blaise attraverso tre fasi:

- *Search process*
- *Select process*
- *Display process*

In tutte e tre le fasi, gli abbinamenti tra i trigrammi della descrizione da codificare e quelli delle descrizioni del dizionario elaborabile (*look-up file*) vengono valutati attraverso una serie di algoritmi i cui risultati vengono confrontati con appositi parametri soglia.
Durante la prima fase (*Search process),* ovvero nella selezione dei trigrammi significativi, Blaise effettua un test che tiene conto del numero dei record contenuti nel *look-up file* e della frequenza del trigramma all'interno del dizionario. Un trigramma allora è considerato significativo, ai fini del *match,* se la sua frequenza non supera il 20 per cento della dimensione totale del *look-up file*.

Nella fase di *Select process* viene calcolata una soglia che discrimina quali dei record che, avendo superato il primo test, debbano essere proposti in visualizzazione. Questa soglia viene calcolata tenendo conto di alcune costanti numeriche e di due valori empirici. Relativamente alla costante numerica che riguarda il numero minimo di trigrammi richiesto in comune tra i due testi, Blaise per *default* pone: *TrigramScoreTreshold=2.*
Nell'ultima fase (*Display process)* infine i record selezionati, all'interno del dizionario informatizzato, vengono visualizzati in un ordine per cui ai primi posti figurano i record con le descrizioni più brevi e con il maggior numero di trigrammi in comune con il testo di input da ricercare.
Nel contesto di codifica assistita approntato per la codifica della Professione non sono state effettuate modifiche alle soglie proposte per *default* da Blaise.
L'unica variazione ha riguardato il parametro ***TrigramScoreTreshold*** che è stato posto uguale a **4**.
Durante la fase di addestramento e studio del contesto di codifica assistita per la variabile Professione si è ritenuto, infatti, che la richiesta di soli 2 trigrammi in comune tra il testo di input e quelli presenti nel dizionario non fosse da ritenere sufficiente per ottenere un *match* significativo.
Le schermate di *look-up* che si aprivano, nella fase di ricerca del testo, rischiavano infatti di proporre troppe opzioni, alcune delle quali risultavano non significative ai fini della codifica. Si è arrivati a questa conclusione dopo aver effettuato un test che ha permesso di valutare anche la performance dell'intero modulo di codifica assistita predisposto nella fase di addestramento.
La modifica di questo parametro deve essere effettuata nella schermata tramite la quale si accede ad una serie di parametri che possono incidere sul "funzionamento" del *DATAMODEL* nel quale viene richiamata la funzione di codifica assistita (per l'approfondimento su questo tema si rimanda al Manuale di Blaise).

Figura 39 - Schermata per la modifica della TrigramScoreTreshold



In generale, si ritiene quindi che la modifica del parametro ***TrigramScoreTreshold*** sia opportuna nel caso di applicazioni caratterizzate da dizionari elaborabili piuttosto corposi e che contengono descrizioni non eccessivamente brevi.
Sempre in funzione dell'ampiezza del dizionario elaborabile (in termini di numerosità dei testi) e della lunghezza delle descrizioni (che spesso si compongono di più parole) può essere opportuno effettuare preventivamente una ***normalizzazione dei testi***.

Questa attività deve, eventualmente, essere effettuata sia sulle descrizioni da includere nel dizionario elaborabile che sulle risposte testuali da codificare e può rendersi necessaria per due motivi:

- principalmente per evitare match tra trigrammi non significativi e limitare quindi il numero di testi del dizionario di cui Blaise farebbe il display, tra i quali il codificatore dovrebbe selezionare quello uguale o più vicino al testo da codificare;
- limitare la digitazione di caratteri testuali per il codificatore (questo aspetto è particolarmente significativo qualora l'attività di codifica sia effettuata nel corso di una rilevazione telefonica).

Gli interventi messi a punto per l'implementazione dell'applicazione sulla codifica assistita della Professione hanno pertanto determinato:

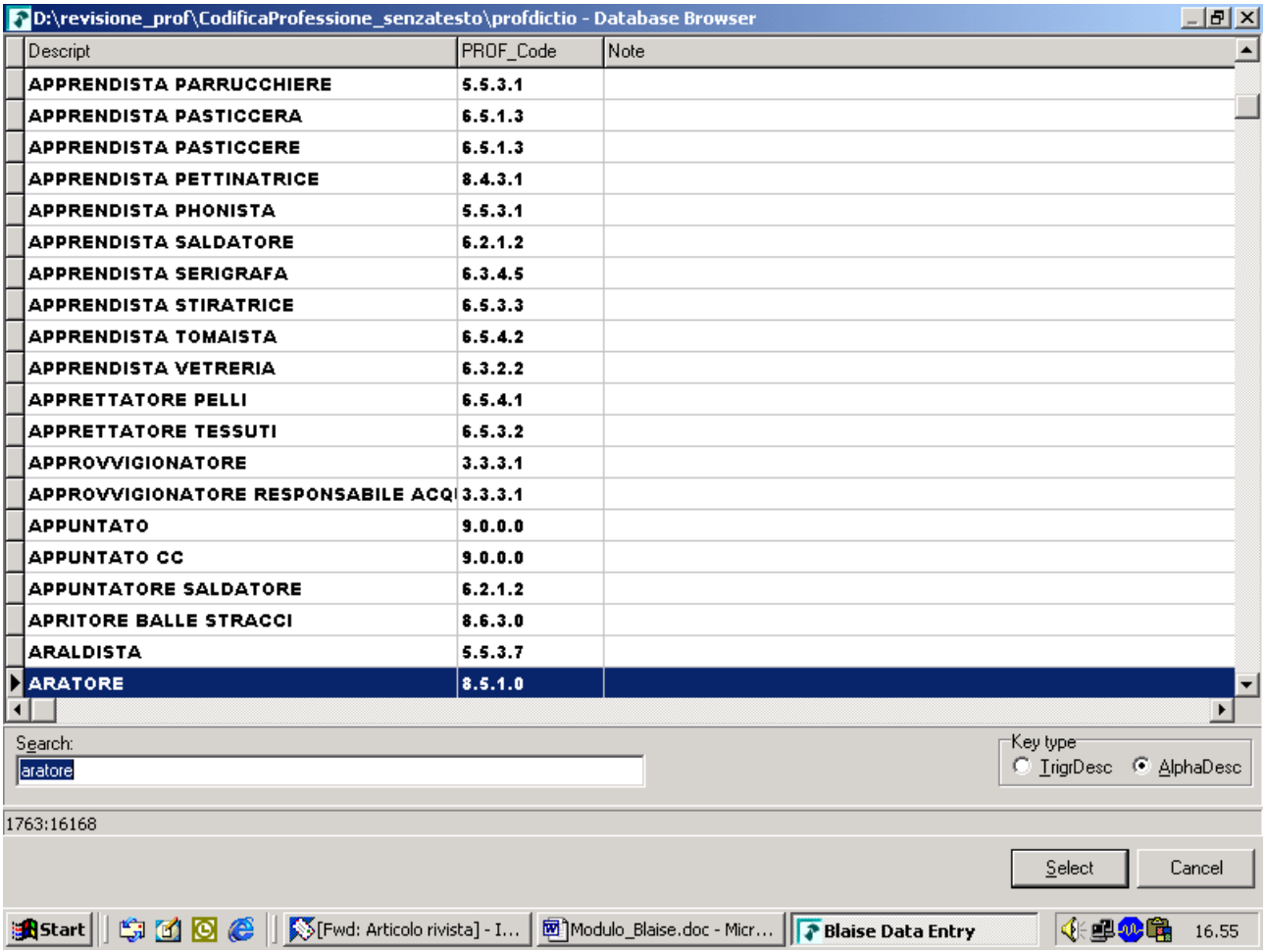
- l'eliminazione all'interno dei testi di tutte le parti invariabili del discorso (preposizioni semplici, articolate, congiunzioni e avverbi);
- l'inserimento delle professioni al genere maschile singolare;
- il non inserimento delle parti variabili del discorso come per esempio i verbi e gli articoli;
- l'eliminazione dai testi della punteggiatura, dei segni ortografici (parentesi, virgolette, trattino, *, $, /, apostrofo);
- il non utilizzo di abbreviazioni nei testi;
- la correttezza ortografica all'interno dei testi.

Infine, sempre nel caso di dizionari elaborabili molto ampi, può essere opportuno ***tenere sotto controllo la frequenza dei trigrammi*** che compongono le descrizioni da includere nel *look-up file*.
Relativamente a questo ultimo punto è interessante infatti osservare come la frequenza eccessiva dei trigrammi, all'interno del *look-up file*, possa determinare un *match* fallito. Ricordiamo ancora che nella fase *Search process* per individuare i trigrammi significativi, il sistema Blaise effettua un test che tiene conto del numero dei record contenuti nel dizionario informatizzato e della frequenza del trigramma all'interno del dizionario stesso. Un trigramma allora è considerato significativo, ai fini del match, se la sua frequenza non supera il 20 per cento della dimensione totale del *look-up file*.
Supponiamo per esempio di voler ricercare la Voce professionale "*Aratore"* propria della Classificazione ufficiale delle Professioni e quindi inserita all'interno del dizionario informatizzato.
Questa parola determina: *NrOfTrigram*=7 → *Working String* di lunghezza L=7+2=9.
La scansione della parola in triplette crea un Array di Trigram T(1,7) dei W(9) caratteri della Working String.
Per capire meglio come lavora l'algoritmo di Blaise è stata calcolata allora la frequenza dei trigrammi in cui viene scomposta la parola *Aratore* all'interno del *look-up file*, come riportato nella tabella che segue.

Tabella 3 – Frequenze dei trigrammi della parola "Aratore"

Qualora in fase di ricerca venisse cambiata la chiave di ricerca (*Key type)* da *TrigDesc* in *AlphaDesc*, cioè venisse utilizzata come chiave di ricerca *l'alfabetica pura* (dove il *matching* viene effettuato secondo l'ordine alfabetico) piuttosto che quella che utilizza come algoritmo di ricerca i trigrammi, si potrebbe ottenere il match cercato come mostra la schermata successiva

Figura 41 - Schermata per la codifica del testo "Aratore" (match alfabetico puro)



La mancata codifica della professione di *Aratore* all'interno del modulo di codifica assistita che utilizza la ricerca per trigrammi resta comunque un caso isolato che non ha preoccupato particolarmente in fase di addestramento in quanto, nello specifico, il *match* fallito è dovuto alla concomitanza di più eventi rari, difficilmente riproducibili ovvero:

- 4 trigrammi su 7 presentavano una frequenza maggiore del 20 per cento della dimensione del dizionario;
- il parametro *TrigramTreshold* è uguale a 4, mentre nel caso specifico della ricerca della parola *Aratore* il numero di trigrammi significativi rimasti è uguale a 3;
- il testo da ricercare è costituito da una sola parola.

Segue che nell'implementazione dei dizionari con i testi opportunamente standardizzati, si è anche prestata attenzione ad:

- ampliare il dizionario con testi che prevedono trigrammi diversi;
- non inserire testi troppo lunghi;
- porre attenzione a non utilizzare, all'interno delle descrizioni empiriche, parole che contengono trigrammi troppo frequenti.

6.3 Verifica delle performance dell'applicazione di codifica

Prima di mettere in produzione un'applicazione molto complessa è opportuno effettuare una verifica delle performance dell'applicazione stessa in termini di successo del sistema nel proporre al codificatore testi coerenti con quello da codificare e di esaustività del dizionario elaborabile rispetto alla tipologia di risposte caratteristiche dell'Indagine per la quale l'applicazione dovrà essere utilizzata.
Al fine di dare un'indicazione su come questa verifica possa essere organizzata, si riportano di seguito gli aspetti salienti del test effettuato sull'applicazione delle Professioni, finora utilizzata in questo manuale a fini esemplificativi.
Il test è stato effettuato utilizzando come base un campione di risposte testuali al quesito sulla Professione rilevato nell'ambito dell'*Indagine sulle Forze di lavoro* (Primo trimestre 1998, per un totale di 86.691). L'analisi di qualità è stata effettuata però solo sui lemmi più frequenti presenti all'interno del file di input. Dagli 86.691 record sono stati cioè estratti dei lemmi appartenenti a classi di frequenza da 51 a più di 1000 occorrenze. Il file di input ottenuto, formato da 210 record è stato poi sottomesso al passaggio di codifica assistita.
Si è ritenuto necessario premettere un lavoro di analisi sul contenuto dei 210 testi da sottoporre a codifica, per capire se l'eventuale insuccesso del match fosse da attribuire alla professione rilevata male o al modulo di codifica assistita carente. Da questa prima analisi è emerso quanto riportato nella seguente tabella:

Tabella 4 – Analisi contenuto testi

Contenuto testi	**Numero record**
Contenuto sufficiente per l'attribuzione univoca del codice (codificabili)	147
Contenuto ambiguo per l'attribuzione univoca del codice (ambigui)	61
Non codificabili	2
Totale record	210

Relativamente a queste casistiche, i risultati di codifica assistita ottenuti sono stati i seguenti:

Tabella 5 – Risultati dei match

Contenuto testi	Proposti in prima posizione	Proposti tra la II e la V posizione	Lista completa	Lista non completa
Codificabili	127	20		
Ambigui			44	17

La prima riga della tabella 5 ci ha portato a concludere che il codificatore sarebbe sempre in grado di attribuire il codice ai testi giudicati "codificabili" in base all'analisi preventiva, in quanto il sistema propone il codice corretto sempre tra le prime 5 righe della schermata di *lookup* e non gli richiede quindi di scorrere più schermate per l'individuazione del codice, cosa che, oltre a richiedere un dispendio di tempo, aumenta il rischio di errore. Nel caso di testi ambigui, per 44 di essi il risultato fornito da Blaise è corretto, in quanto il sistema fornisce una lista esaustiva di casi coerenti con quello da codificare. Per gli altri 17 casi ambigui, invece, la lista dei codici proposta da Blaise non è risultata essere completa, per cui è stato necessario procedere con un ulteriore arricchimento del dizionario elaborabile.
La conclusione a cui si è pervenuti a seguito del test, comunque, è stata che l'applicazione poteva essere messa in produzione, in quanto nel 91 per cento dei testi analizzati il sistema ha prodotto un match di codifica corretto (ossia: codice corretto in prima posizione, oppure compreso tra la seconda e la quinta riga, comunque la lista di codici proposta è risultata completa).

ALLEGATO A – generazione della Library per la definizione dell'albero della classificazione (<u>ATECOalbero TXT LIB.man)</u>

```
SETTINGS
  AUTOMOVE = NO
```

```
USES
  DATAMODEL inputdata   {structure of input file }
  FIELDS
    code        : STRING[10]
    name        : STRING[10]
    description : STRING[200]
  ENDMODEL
  DATAMODEL Blaiselib   {structure of output file }
  FIELDS
    line : STRING[200]
  ENDMODEL
```

```
INPUTFILE          { ... input file ... }
  inf : inputdata ('G:\Blaise\sorgenti\ATECO\dizionari\AtecoAlbero.BLAISE.csv', ASCII)
  SETTINGS
    SEPARATOR=';'
OUTPUTFILE         { ... output file ... }
```

```
  ENDDO
  outf.WRITE
  line := '    HIERARCHY' outf.WRITE
ENDIF
length := LEN(code)
count := count + 1
IF (length = prevlength) THEN                { Code length is the same so we are still on the same level}
  level := prevlevel
  IF level > 1 THEN                          { lookup if all digits of levels 1..LEVEL-1 are the same and determin code of the deepest level}
    lw := levelwidth[level-1]
    IF (SUBSTRING(code,1,lw) <> SUBSTRING(prevcode,1,lw)) THEN
      DISPLAY('code '+prevcode+' followed by code '+code)
      HALT
    ENDIF
  ELSE
    lw := 0
  ENDIF

  IF (SUBSTRING(code,lw+1,length-lw) = SUBSTRING(prevcode,lw+1,length-lw)) THEN      { check if lasts digits are new}
    DISPLAY('code '+prevcode+' followed by '+code)
    HALT
  ENDIF

  line := prevline + ','                     { if levels are the same just add a comma}
ELSE
  IF (length > prevlength) THEN
    level           := prevlevel + 1         { We nest one level deeper. It is impossible to nest more than one level at a time}
    levelwidth[level] := length
    line            := prevline + ' =('
  ELSE { length < prevlength }
    level           := 1                     { Determine number of levels to go back to}
    WHILE (levelwidth[level] < length) do
      level := level+1
    ENDWHILE

    IF (levelwidth[level] <> length) THEN
      DISPLAY('length of the code '+code+' is not expected here')
      HALT
    ENDIF { lookup if all digits of levels 1..LEVEL-1 are the same and determin code of the deepest level}

    IF level > 1 THEN
      lw := levelwidth[level-1]
      IF (SUBSTRING(code,1,lw) <> SUBSTRING(prevcode,1,lw)) THEN
        DISPLAY('code '+prevcode+' followed by code '+code)
        HALT
      ENDIF
    ELSE
      lw := 0
    ENDIF { voeg sluithaken toe }
    line := prevline + '  '
    FOR i := 1 TO prevlevel-level DO
      line := line + ')'
    ENDDO
    line := line + ','
  ENDIF
ENDIF  { bepaal digits laatste nivo }
IF (level > 1 ) THEN
  lw := levelwidth[level-1]
ELSE
  lw := 0
ENDIF
digit := SUBSTRING(code, lw+1, length-lw)
IF (SUBSTRING(digit,1,1)='.') THEN
  digit := substring(digit, 2, LEN(digit)-1)
ENDIF
curline := name +'  ('+digit+')  "'+description+'"'
curline := '{'+code+'}       '+curline
prevcode   := code
prevline   := curline
prevlevel  := level
prevlength := length
```

```
 IF (count > 1) THEN
   outf.WRITE
 ENDIF
IF inf.lastrecord THEN
   level := 1
   line := prevline + '  '
   FOR i := 1 TO prevlevel-level DO
     line := line + ')'
   ENDDO
   outf.WRITE
   line := '    ENDCLASSIFICATION'   outf.WRITE
   line := 'ENDMODEL'                outf.WRITE
 ENDIF
```

Bibliografia

ACTR V3. *User Guide*. Statistics. Canada: 2002.

ACTR V3. *Parser Guide.* Statistics. Canada: 2002.

Appel M. and Hellerman E. *Census Bureau experience with Automated Industry and Occupation Coding*. In American Statistical Association, 1983. (Proceedings of Section on Survey Research Methods, pages 32-40).

BLAISE a survey processing system. *Developer's Guide*. Statistics Netherlands, 2002.

Chen B., and R. Creecy, and M. Appel. *Error control of automated industry and occupation coding.* Journal of Official Statistics, 1993. (vol. 9 729-745).

De Angelis, Roberta, e Stefania Macchia, e Loredana Mazza. A*pplicazioni sperimentali della codifica automatica: analisi di qualità e confronto con la codifica manuale.* Roma: Istat, 2000. (Quaderni di ricerca – Rivista di statistica ufficiale, n.1, 29-54).

Istat. *Classificazione delle attività economiche.* Roma: Istat, 1991. (Metodi e norme, serie C-11).

Lyberg and Dean. *Automated Coding of Survey Responses: an international review*. Washington: 1992. (DC Conference of European Statisticians, work session on statistical data editing, 1992).

Kalpic D. *Automated coding of census data.* 1994. (Journal of Official Statistics, vol. 10, 449-463).

Knaus R. *Methods and problems in coding natural language survey data.* 1987. (Journal of Official Statistics, vol. 1, 45-67).

Macchia, Stefania. *Sperimentazione, implementazione e gestione dell'ambiente di codifica automatica della Classificazione delle attività economiche.* Roma: Istat, 2002. (Documenti Istat, n.2 2/2002).

Macchia, Stefania. *Integration of sources to build a dictionary for Automated Coding of Industry.* Palermo: 2001. (In CLADAG Conference, 5-6 luglio 2001).

Macchia, Stefania and Marcello, D'Orazio. *Analysis of Textual data for integrating an automated coding environment system and building a system to monitor the quality of its results*. Lausanne, Switzerland: 2000. (5th Journées internationales d'analyse statistique des données textuelles, 9-11 Mars 2000).

Massingham R. *Data capture and Coding for the 2001 Great Britain Census*. Hull, Canada: 1997. (In XIV Annual international symposium on methodology issues, 5-7 November 1997).

Touringny J.Y. and Moloney. *The 1991 Canadain Census of Population experience with automated coding, statistical data editing, 2.* New York: 1995. (United nations statistical commission, 1995).

Wenzowski M.J. *A Generalised Automated Coding System.* ACTR: 1998. (Survey Methodology, vol. 14 299-308).

Touringny J.Y. and Moloney. *The 1991 Canadain Census of Population experience with automated coding, Statistical Data Editing, 2.* New York: 1995. (United nations statistical commission, 1995).